**University of Alberta**

# Motion Similarity Analysis and Evaluation of Motion Capture Data

by

**Tong Guan**

A thesis submitted to the Faculty of Graduate Studies and Research in partial fulfillment of the requirements for the degree of **Master of Science**.

Department of Computing Science

Edmonton, Alberta
Fall 2004

Library and Archives Canada

Published Heritage Branch

395 Wellington Street
Ottawa ON K1A 0N4
Canada

Bibliothèque et Archives Canada

Direction du Patrimoine de l'édition

395, rue Wellington
Ottawa ON K1A 0N4
Canada

*Your file Votre référence*
*ISBN: 0-612-95757-8*
*Our file Notre référence*
*ISBN: 0-612-95757-8*

The author has granted a non-exclusive license allowing the Library and Archives Canada to reproduce, loan, distribute or sell copies of this thesis in microform, paper or electronic formats.

The author retains ownership of the copyright in this thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without the author's permission.

L'auteur a accordé une licence non exclusive permettant à la Bibliothèque et Archives Canada de reproduire, prêter, distribuer ou vendre des copies de cette thèse sous la forme de microfiche/film, de reproduction sur papier ou sur format électronique.

L'auteur conserve la propriété du droit d'auteur qui protège cette thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou aturement reproduits sans son autorisation.

---

In compliance with the Canadian Privacy Act some supporting forms may have been removed from this thesis.

While these forms may be included in the document page count, their removal does not represent any loss of content from the thesis.

Conformément à la loi canadienne sur la protection de la vie privée, quelques formulaires secondaires ont été enlevés de cette thèse.

Bien que ces formulaires aient inclus dans la pagination, il n'y aura aucun contenu manquant.

Canada

# Acknowledgements

I extend my sincere thanks to my supervisor Dr. Yee-Hong Yang for his guidance and support during the entire research process, as well as his helpful suggestions and patient editing of this thesis.

I thank the members of my examining committee: Dr. Pierre Boulanger and Dr. Yau Shu Wong for taking the time to read and comment on this thesis.

I would like to express my appreciation to every member of the Computer Graphics Research Group, Dr. Minglun Gong, Xuejie Qin, Hai Mao, Daniel Neilson, Yi Xu, Jiayuan Zhu, Cheng Lei, and Nathan Funk for their inspiring discussions and ideas.

I am also grateful to the Department of Computing Science, and the University of Alberta for their financial support during my master program.

Additionally, I want to acknowledge Alias|Wavefront for Maya software, the Movement Laboratory of Stanford University for their motion capture data distributed on the web.

Finally, many thanks to my parents and my husband for their support and love.

# Contents


**1 Introduction** 1
- 1.1 Motivation and Objective . . . 1
- 1.2 Outline . . . 3

**2 Related Work** 4
- 2.1 Motion Capture . . . 4
- 2.2 Motion Capture Data Representation . . . 5
- 2.3 Interpolation Synthesis . . . 7
  - 2.3.1 Multi-linear Interpolation . . . 8
  - 2.3.2 Verbs and Adverbs Mechanism . . . 10
  - 2.3.3 Interpolation by Multi-Resolution Analysis . . . 13
  - 2.3.4 Interpolation by Fourier Expansion . . . 17
  - 2.3.5 Summary of Interpolation Synthesis . . . 18
- 2.4 Reassembling Synthesis . . . 19
  - 2.4.1 One-level Motion Graph . . . 21
  - 2.4.2 Two-level Motion Graph . . . 24
  - 2.4.3 Hierarchical Motion Graph . . . 28
  - 2.4.4 Summary of Reassembling Synthesis . . . 34
- 2.5 Motion Transition Mechanisms . . . 35
  - 2.5.1 Candidate Transition Point Selection . . . 35
  - 2.5.2 Transition by Using Spacetime Constraints . . . 36
  - 2.5.3 Transition by Stitching . . . 37
  - 2.5.4 Transition by Interpolation . . . 39
- 2.6 Motion Similarity Analysis . . . 40
  - 2.6.1 Similarity Based on Joint Attributes . . . 40
  - 2.6.2 Similarity Based on Point Clouds . . . 43
  - 2.6.3 Joint Weight . . . 47

**3 A Novel Approach for Motion Similarity Analysis** 48
- 3.1 Feature Selection . . . 48
- 3.2 Similarity Analysis . . . 52
  - 3.2.1 $D_{rp}$ Computation . . . 52
  - 3.2.2 Curvature Cross Correlation . . . 53

**4 Experimental Results** **59**
4.1 Run Time Comparison . . . . . 59
4.2 Accuracy Study . . . . . 60
4.2.1 Accuracy Criteria . . . . . 60
4.2.2 Optimal Attribute Weights Computation . . . . . 61
4.2.3 Results for Accuracy Evaluation . . . . . 63
4.3 Visual Acceptability . . . . . 68

**5 Conclusions and Future Work** **73**
5.1 Contributions . . . . . 73
5.2 Future Work . . . . . 75

**Bibliography** **76**

**A The Header Part of a BVH File** **78**

**B Joint Orientation Representations and Conversions** **79**

# List of Tables

2.1 The optimal joint weight set in [21]. Only the joints with non-zero weights are shown. . . . . 47

4.1 Run time to compute the similarity between two motion clips for different motion similarity analysis methods. . . . . 60
4.2 Comparison between different approaches for motion similarity analysis. 71

# List of Figures


2.1 The hierarchical structure defined in the header of a BVH file. . . . 6
2.2 Calculating joint position in global coordinates. . . . . 7
2.3 Left: Initial hand position data denoted by "+." Right: Re-sampled hand position data denoted by circles. Note that the 3D hand positions are illustrated with 2D points. . . . . 8
2.4 Linear interpolation. . . . . 9
2.5 Mapping between clock time $T$ and generic time $t$ by keytimes. $NumKeyTimes$ is 4. At clock time $T_A$ and $T_B$, motion A and motion B have similar movements. . . . . 11
2.6 Top: the optimal path computed by dynamic programming. Bottom: the warping process based on the optimal path. . . . . 15
2.7 The structure of a one-level motion graph. . . . . 21
2.8 The problematic nodes in the graph. Node 5 is a dead node. Node 4 is a sink node. . . . . 22
2.9 The structure of a two-level motion graph. The lower level encodes source motion data and their possible transitions on a frame-by-frame basis. The higher level records the clusters and their connections. . . 24
2.10 The process for building cluster trees. First row: Original motion data on a frame-by-frame basis. Each node is a frame, and each edge between two nodes means that one frame can be seamlessly followed by the second frame. Second row: Transitions denoted by arrows in the lower layer are generated. Third row: Similar frames are grouped into clusters. Last row: Two cluster trees are formed for the 2 frames marked with red circles. (This figure is taken from the original paper.) 26
2.11 A valid path and its corresponding motion sequence. The path involves 6 edges and 4 clips. The first edge and the last edge are, respectively, used to mark the starting and end point of the path. If $e_4$ is replaced with $e$, the path becomes invalid because $fromFrame(e)$ appears before $toFrame(e_3)$ in time, the second condition is not satisfied. . . . . 30
2.12 The clustering process for building top level graph $G_1$. Every edge between two nodes is represented as an entry in a matrix. Similar frame pairs in a neighborhood form a cluster (marked with a blue circle). The center (marked with a red star) of these edges becomes one of the edges of $G_1$ . . . . . 31

2.13 An edge at the top level graph is represented by a binary tree. . . . 32
2.14 Transition by stitching: a quadratic smoothing function controls the amount of displacement of two connected clips in a smoothing window. Left top: The magnitude of discontinuity between two clips. Left bottom: The smoothing function. Right: The resulting transition after smoothing. (This figure is adopted from the original paper.) 38
2.15 Transition by linear interpolation. $t_s$ and $t_e$ are the start transition point and end transition point, respectively. . . . . 39
2.16 Similarity analysis based on joint angles with different attribute weight sets between left and right (columns correspond to frames from the first clip, rows correspond to frames from the second clip). Top: The distance matrix. Bottom: The resulting similar frame pairs are denoted by blue dots. . . . . 42
2.17 Similarity analysis based on joint velocities and accelerations in different datasets. Left: The dataset has only 2 clips. Right: The dataset has 25 clips. Top: The distance matrix. Bottom: The resulting similar frame pairs are denoted by blue dots. . . . . 44
2.18 Top row: Two motion clips on the different level. Middle row and bottom row: Similarity analysis based on point clouds. Left: Two clips on the same level. Right: Two clips on the different level. Middle: The distance matrix. Bottom: The resulting similar frames. . . 46

3.1 Two space curves formed by the movement of the LeftLowArm joint in two walking sequences. . . . . 49
3.2 The Frenet frame of space curve . . . . . 50
3.3 Left: Two spiral curves. Middle: Curvature and torsion for Curve A. Right: Curvature and torsion for Curve B. . . . . 51
3.4 Motion similarity analysis process. . . . . 52
3.5 $D_{rp}$ computation between walking and running. Left: $D_{rp}$ matrix. Right: candidate similar frame pairs. . . . . 53
3.6 B-spline approximation between two control points $P_i$ and $P_{i+1}$. . . 55
3.7 Left: The kernel used in Gaussian smoothing. ($\sigma = 2$ , $size = 9$) Right: The curvature of the motion curve for the joint leftLowArm in a walking clip, where the top signal is the curvature (shifted up by 3) calculated with a Gaussian filter, and the bottom signal is the curvature calculated without a Gaussian filter. . . . . 56
3.8 Cross correlation operated on the feature vectors of two motion clips. One dimensional feature formed by the trajectories of the left shoulder between a running and a walking sequence. . . . . 56
3.9 Left: Cross correlation results. Right: Similar frame pairs. . . . . 58

4.1 Two motion clips on the same floor plane. . . . . 63

4.2 Two motion clips on different levels. Top: Running on the floor plane. Bottom: Walking along a ramp with a slope of 15 degrees. $H$ represents the vertical distance by which the walking clip is shifted down from the floor plane. . . . . 63
4.3 Similarity analysis based on point clouds. (a) two clips are on the same floor plane; (b)(c)(d) on 3 different level cases. Top: The root trails of walking and running. The dotted boxes mark the areas with a smaller vertical distance of the root joints. Middle: The distance matrix (darker areas imply a lower distance measure). Bottom: The resulting similar frame pairs denoted by blue dots. The rows represent the frames in the running clip, while the columns represent the frames in the walking clip. . . . . 64
4.4 Similarity analysis based on: (a) Curvature and $D_{rp}$; (b) Joint angles and velocities; (c) Joint positions and velocities. Top left: Cross-correlation results. Top middle and right: Distance matrix. Bottom: The resulting similar frame pairs. . . . . 65
4.5 Performance comparison between the same level case and the 3 different level cases. In the different level case, the performance for the point clouds approach decreases largely when compared with its performance in the same level case; while the other approaches perform almost the same in both cases. . . . . 66
4.6 Similarity analysis based on joint angles and joint velocities with different attribute weight sets. Left: A user specified attribute weight set. Middle and right: Two optimal attribute weight sets. Top: The distance matrix. Bottom: The resulting similar frame pairs denoted by blue dots. . . . . 67
4.7 Performance comparison for the joint angle approach by using different attribute weight sets. The performance of the joint angle approach is largely improved by using the computed optimal attribute weight sets according to $D_p$. . . . . 68
4.8 Motion similarity analysis results for Dataset B. (a) the cross correlation results of our approach; (b)(c)(d) the distance matrices; (e)(f)(g)(h) the resulting similar frame pairs. . . . . 69
4.9 Performance comparison between 3 different datasets. Top row: Dataset A, a walking and a running ($300 \times 236$ frames). Middle row: Dataset B, a rocknroll and a highwire walking ($260 \times 300$ frames). Bottom row: Dataset C, 12 different motion clips. (a) the position difference; (b) the velocity difference; (c) the acceleration difference. . . . . 70
4.10 Transitions between walking (green) and running (red). Left: Skeleton poses at the selected similar frame pairs. Middle: Transitions without linear interpolation. Right: Transitions with linear interpolation. The screen shots are down-sampled by a factor of 3. . . . . 72

# Chapter 1

# Introduction

## 1.1 Motivation and Objective

In computer graphics, character animation is an important research area and many research works focus on creating realistic and natural human animations. Recently, the price of motion capture hardware has dropped significantly, and hence, techniques based on motion capture data have become commonly used alternatives to the other two animation techniques, namely, keyframing and physical-based simulation. Because of the ease of use, realistic and natural motion capture data have become widely used in commercial applications; in particular, with the development of practical motion capture and editing techniques [4, 5, 6, 14, 16, 18, 20, 22].

To generate new motion patterns by reusing existing motion capture data is still an interesting open problem. Two major approaches exist. One is to use motion blending by combining two or more motion examples to form a new motion clip [16, 18, 20, 22]. The other approach is motivated by the video texture technique [2, 11, 12, 13], by which a new motion sequence is generated by stitching the original motion clips in a new order. Because it directly reuses the source motion sequences, it is able to preserve realism and high-level details of the original motion. Moreover, motion blending focuses on creating an individual motion clip, while motion reassembling focuses on generating a new motion sequence. Both of the approaches mentioned above have the same goal to create new motions from exiting motion capture data. In this thesis, the focus is on techniques to stitch the original clips in a new order,

in particular, in locating the best transition points in connecting two clips.

To use the raw motion capture data directly is difficult because of the unstructured and high complexity nature inherent in the data. Thus, motion analysis tools, especially for analyzing human body motion, are indispensable and have become a very important research topic in the context of motion editing. Motion similarity analysis provides the foundation for many recent research works. For example, in [2, 11, 12, 13], similar frames are detected first to find candidate transition points. Then a new motion sequence is synthesized by reordering motion clips. As discontinuities are introduced at transition points, they must be carefully selected. Thus, a smooth transition between clips is generated. While in motion blending [4, 16, 18, 20, 22], time warping is performed to align the example motions in the time domain (synchronization) according to the motion similarity.

Some motion editing techniques require the user to determine the similarity manually [18, 20]. The animator specifies similar motion frames according to his/her perception and experience. Therefore, the quality of the resulting motion depends significantly on the animator's skills. Moreover, it is a labor-intensive process. In order to reduce the burden on animators and to increase the speed of motion similarity analysis, some methods [2, 11, 12, 13] have been proposed to automatically detect motion similarity.

Two main types of motion similarity metrics based on different motion features are defined in previous works. The first one is based on joint orientations and velocities [2, 13], and the other on the distance between sample points (point clouds) [12]. Both of them emphasize on the pose similarity of two frames. In order to capture kinematic information, such as velocity and acceleration, the first approach directly incorporates these features, while the second approach considers the difference between the neighborhoods of the two frames. For the first approach, as the motion capture data are directly represented by the joint angle of a skeleton, the similarity metrics based on joint orientations can be easily computed directly. The disadvantage of this technique is that it uses a weighted sum of multiple joint attributes as a measure. The optimal attribute weights are very difficult to identify and may be dependent on motion patterns. As a result, the dynamic information of the source data may not be incorporated well. The second approach is very time consuming

and dependent on the coordinate system. The assumption that a motion is not changed by a rigid 2D transformation restricts its application to the movements on the same ground plane.

In order to tackle the problems mentioned above, the present work introduces two novel similarity features to reuse realistic motion capture data. Curvature, one of the intrinsic properties of space curve, is used to capture the kinematic information. The difference between joint positions in their own parent coordinates is used to evaluate the similarity of body configurations. Both of these two features are coordinate invariant and can be computed efficiently. Most importantly, no attribute weight is needed in our approach. Additionally, to evaluation the performance of different motion similarity methods, this thesis also presents a general criterion that helps to determine optimal attribute weights used in other approaches. The experimental results demonstrate that our approach for motion similarity analysis can generate visually acceptable results and that the other methods' performance can be improved by using our new evaluation criterion.

## 1.2 Outline

This thesis is organized as follows. Chapter 2 concerns the related previous works. In particular, motion capture data representation, motion synthesis by reusing motion capture data, and motion similarity analysis are discussed. The new motion similarity analysis is presented in Chapter 3. Chapter 4 shows the experimental results to compare the new approach with other approaches. In Chapter 5, conclusion and future work are discussed.

# Chapter 2

# Related Work

## 2.1 Motion Capture

Realistic articulated figure animations have played an important part in a variety of applications including advertising, entertainment, education and simulation. Though there are many animation techniques in this research area, to generate natural and realistic animation for a virtual human body is still a challenging task. There are several reasons for this. First, we are very familiar with human movements. It is quite easy for a person to identify any unnatural artifacts. Another reason is the complex structure of a human body. This complexity makes it difficult to mathematically define human motions and accurately control an articulated character in a virtual environment as well as to model their interactions. Finally it is the fine details inherent in the human motions that makes the animation of human body so challenging. Thus, to provide a practical solution is still much desired.

Motion capture is a standard character animation technique, by which the movement of a live subject is recorded and then mapped onto a computer generated graphical model. Both facial motion and body motion can be captured. In this thesis, the focus is on full-body human motions. The major benefit of this method is that the motion capture data contain all the high-level details of live motion, which reveal the personality or mood of the recorded live subjects. On the other hands, some disadvantages exist in this technique. First, except for planning the capture session, animators have little control over the generated animations. Moreover, the

motion capture data are difficult to edit. Typically motion capture data record the pose in each frame of the motion, not just important instants of the movements as in the case of keyframing. Thus, editing means a lot of data need to be dealt with.

Recently with the improvement of motion capture techniques and the lowering cost of the hardware, realistic and natural motion data have become very popular. For example, an animator can easily acquire a highly detailed motion clip with the desired motion pattern, and then use it to drive a graphical character to produce a stream of convincing or visually appealing animation. Indeed, in the recent movie, "The Lord of the Rings: The Two Towers," the character Gollum was animated using this movement retargeting technique and won the MTV Best Digital Performance Award in 2003 [10]. To make use of such abundant motion sources, many research works have been proposed based on directly reusing motion capture data. Typically they are divided into two categories, interpolation-based synthesis and reassembling-based synthesis.

The remainder of this chapter is described as follows. First, the representation of motion capture data is described. Then two categories of motion synthesis algorithms based on exiting motions are discussed, namely, motion interpolation and motion reassembling. As transition techniques are an integral part of motion reassembling, and an application of motion blending, they are discussed next. Since motion similarity analysis is a critical stage for the selection of motion transition points, previous approaches for motion similarity analysis are reviewed and the related problems are identified in the last section of this chapter.

## 2.2 Motion Capture Data Representation

Generally, motion capture data are represented in the form of joint angles plus 6 degrees of freedom for the rotations and translations of the body. In most cases, the BVH file format is used. The name BVH stands for BioVision Hierarchical data. A typical BVH file consists of two parts: a header section and a motion section. Appendix A shows the header section of a standard BVH file. The first part defines the skeleton's hierarchical structure and its initial configuration, which is illustrated

in Figure 2.1. The second part records the motion information.

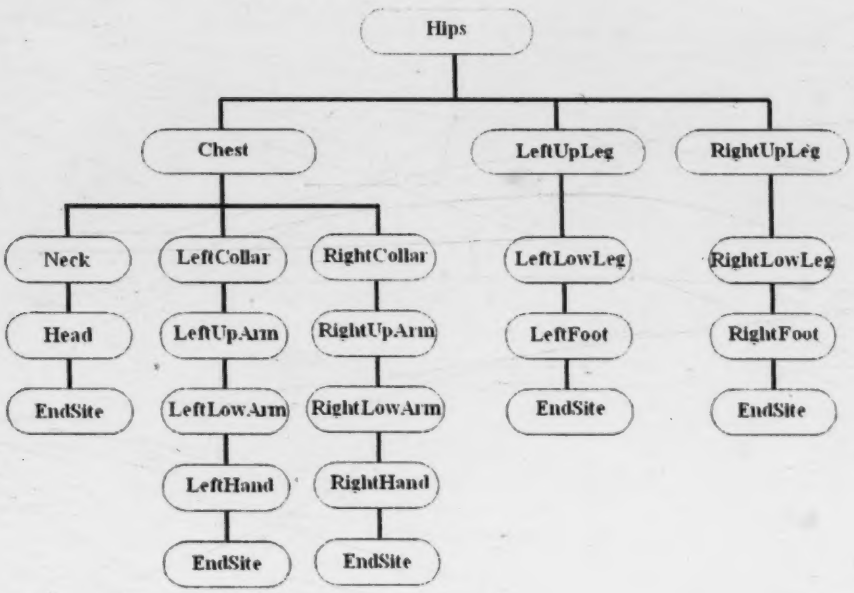



Figure 2.1: The hierarchical structure defined in the header of a BVH file.

In the motion section, each row corresponds to the pose of a frame, each column corresponds to a channel (a degree of freedom). Usually the root joint (hips) has 6 channels, 3 for each axis of translation, 3 for each axis of rotation. While each of the other joints has 3 channels to represent its orientation with respect to its own parent. Thus, all the channels along with the skeleton structure can completely define the configuration of a character at a frame. The global position of a joint can be easily derived from the BVH file. In the following example, the foot position in the global coordinates is calculated according to Equation 2.1.

$$\mathbf{p}_i = T_{Hips} R_{Hips} T_{UpLeg} R_{UpLeg} T_{LowLeg} R_{LowLeg} \mathbf{v}_{foot} \tag{2.1}$$

where $\mathbf{v}_{foot}$ is the offset vector of the foot relative with its parent LowLeg, $T_{Hips}$ and $R_{Hips}$ are, respectively, the translation matrix and rotation matrix of the root joint. $T_{UpLeg}, R_{UpLeg}, T_{LowLeg}$, and $R_{LowLeg}$ are similarly defined for the joint UpLeg and LowLeg.

Compared with the motion data generated by traditional methods like keyframing, motion capture data are difficult to work with. One reason is that every frame

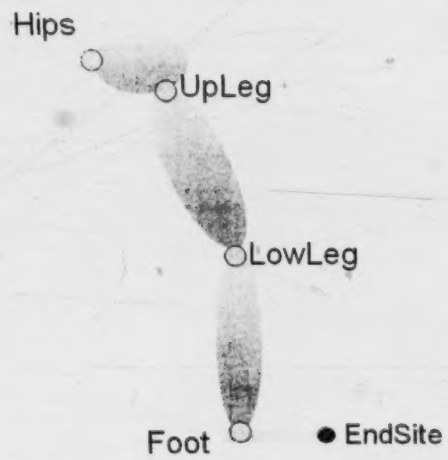



Figure 2.2: Calculating joint position in global coordinates.

of a motion is recorded, and each frame has about 57 channels. Thus, there are a lot of data to be processed. Another reason is that the data provide little clues on the important properties of a motion and the effects on the motion when the properties are changed. As a result, reusing motion capture data has become an open problem in the context of animation.

## 2.3 Interpolation Synthesis

Motion interpolation is one of the typical motion edition techniques, by which new motions are generated by combining two or more motion examples. Usually, a continuous parameterized motion space is constructed from a small set of motion examples by multi-target blending, then the in-between motion can be derived from the parameter space. Generally, motion synthesis is restricted to a small number of parameters because a higher dimensional parameter space requires a larger motion dataset and the cost of storage and computation increases significantly. Currently, many research works belong to this approach. In the remainder of this section, some typical interpolation synthesis schemes are investigated.

### 2.3.1 Multi-linear Interpolation

The researchers in [22] present a very simple but efficient multi-linear interpolation method to generate a new motion clip from a given database. First, according to the specification of the required motion, a subset of motion clips that are most similar to the desired motion are selected. Then the new motion is synthesized by linear interpolating the selected data. The process is illustrated by the following example.

Considering that the hand of an articulated figure is required to reach a particular space position. In this setting, the motion clip is parameterized by the 3 coordinates of the hand's global position. In this example, the motion parameter space is 3 dimensions and is defined by the position that the hand is required to reach. To facilitate searching for the required subset of data, the original motion data are resampled so that the resampled data form a regular grid of the parameter space (see Figure 2.3). As a result, the selection of the required data subset does not require a search. Instead, the motion data at the grid points closed to the required hand position are directly selected as the interpolated dataset.

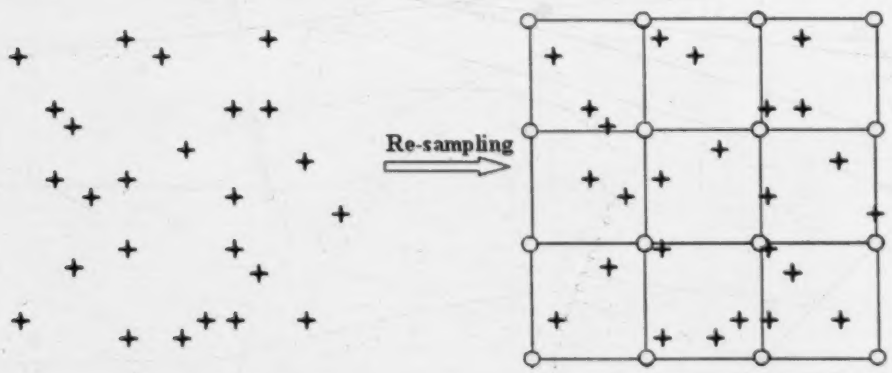



Figure 2.3: Left: Initial hand position data denoted by "+." Right: Re-sampled hand position data denoted by circles. Note that the 3D hand positions are illustrated with 2D points.

After the required motion subset is found, the resulting motion is generated by successive interpolations in each dimension. In Figure 2.4, point $\mathbf{t}$ is the desired

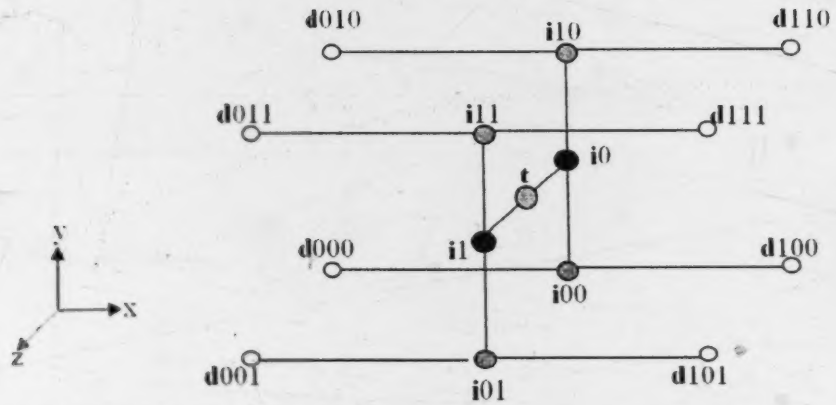



Figure 2.4: Linear interpolation.

hand position, **d000** to **d111** are hand positions of the selected data which are closed to point **t**. First, four interpolations along the x-axis are performed and **i00**, **i01**, **i10** and **i11** are the intermediate interpolation results. **i00** is computed in Equation 2.2, where $u$ is the interpolation factor, which is derived from the desired hand geometrical position relative to the interpolated motion data (Equation 2.3).

$$\mathbf{i00} = (1-u) \times \mathbf{d000} + u \times \mathbf{d100}, \quad u \in [0,1] \tag{2.2}$$

$$u = \frac{\mathbf{t}_x - \mathbf{d000}_x}{\mathbf{d1000}_x - \mathbf{d000}_x} \tag{2.3}$$

Then, two interpolations along the y-axis are performed to the intermediate results generated in the previous step. Similarly, **i0** and **i1** are obtained. Finally one interpolation is performed between **i0** and **i1** along z-axis and the resulting motion is achieved.

Besides the hand-reaching example discussed above, the authors in [22] also present a walking example on different slopes, where one parameter, the slope is used to construct the parameter motion space. As walking clips may have different length in one cycle, a simple synchronization mechanism is applied to the selected motions before interpolation. All the walking clips with the same cycles are uniformly (linearly) resampled to the same length. Thus, all the clips are synchronized on a cycle-by-cycle basis.

This interpolation synthesis requires at least $2^p$ motion data, where $p$ is the number of parameters. In general, for $p$ parameters, $2^p - 1$ interpolations are performed with a window of $2^p$ motion data. A denser motion dataset can produce more accurate resulting motions but it increases the storage and computation requirements.

### 2.3.2 Verbs and Adverbs Mechanism

In [18], C. Rose et al. describe an interpolation method (called verbs/adverbs mechanism) to generate a wide variety of motions from some existing motion examples. In this technique, motions are characterized by emotional contents (happy or tired) or kinematic aspects such as turning left, going uphill or downhill. These parameterized motions are called verbs and the parameters that control them are called adverbs. Radial basis function and low order (linear) polynomials are used to interpolate the verbs in a multidimensional parameter space defined by the adverbs.

In the framework of [18], each joint of an animated character contains one or more rotational degree of freedom. Thus, each motion example is defined by a number of DOF functions. The $j^{th}$ DOF function of the $i^{th}$ motion example $M_i$ is denoted by $\Theta_{ij}(T)$, where $T$ is the clock time or frame index. Each $\Theta_{ij}$ is represented as a uniform cubic B-spline curve specified by $NumCP$ control points as shown in Equation 2.4.

$$\Theta_{ij}(T) = \sum_{k=1}^{NumCP} b_{ijk} B_k(T) \tag{2.4}$$

where $B_k(T)$ is the B-spline, and $b_{ijk}$ is the control point of the B-spline. $i$ is the index of motion example. $j$ is the index of DOF function. $k$ is the index of control point.

Verbs are constructed from sets of similar motions with distinct styles. For example, within a set of walk cycles, they should have similar poses at starting points, ending points, and have the same step number, etc. Their difference only exists in the emotional aspects or kinematic aspects. Then for each motion example, a set of appropriate adverb values are identified manually by the user. And a corresponding point is added into the adverb space. So a verb $M$ is defined by a set

of similar motions in Equation 2.5.

$$M_i = \{\Theta_{ij}(T), \mathbf{p}_i, K_m\} \quad \begin{matrix} i = 1 \ldots NumMotions, \\ j = 1 \ldots NumDOF, \\ m = 0 \ldots NumKeyTimes \end{matrix} \tag{2.5}$$

where $\Theta_{ij}(T)$ is the $j^{th}$ DOF function for the $i^{th}$ motion data $M_i$. $\mathbf{p}_i$ is the location of $M_i$ in the adverb space. $K$ is the set of keytimes which describes the instants when important motion structural elements happen, such as heel-strike and toe-off. Since the motion examples in a verb have similar structure elements, as a result, they should have similar keytimes. In other words, the same important event should happen in all the motion examples of a verb. In the framework presented in [18], the keytime set $K$ is manually detected by the user and is used to synchronize different motions. The synchronization process is called time warping and it is achieved by a piecewise linear mapping. The details are described as follows.

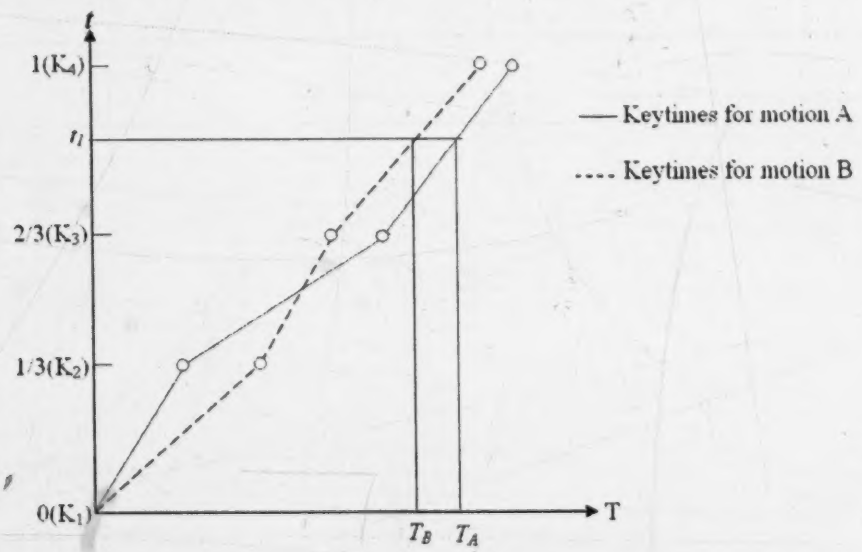



Figure 2.5: Mapping between clock time $T$ and generic time $t$ by keytimes. $NumKeyTimes$ is 4. At clock time $T_A$ and $T_B$, motion A and motion B have similar movements.

In this step, the keytimes are used to define a linear mapping from a clock time $T \in \{0 \ldots K_{NumberKeyTimes}\}$ to a generic time $t \in \{0 \ldots 1\}$. The first keytime of the verb, $K_1$ is mapped to be 0, while the last keytime, $K_{NumberKeyTimes}$ to 1. And the generic time $t$ for a clock time $T$ between 0 and the last keytime $K_{NumKeyTimes}$ is

defined in Equation 2.6.

$$t(T) = \left\{(m-1) + \frac{T - K_m}{K_{m+1} - K_m}\right\} \frac{1}{NumKeyTimes - 1} \quad when\ T \in [K_m, K_{m+1}] \tag{2.6}$$

The mapping process is illustrated in Figure 2.5. Once the motion examples are re-parameterized from the clock time $T$ to the generic time $t$, all the motions and their interpolated results will have similar structure elements for a given $t$. The corresponding B-spline control points for each DOF in each motion will specify similar movement in each motion.

After verbs are constructed with sets of motion examples with its associated keytimes and adverb settings, a new motion with a set of specified adverb values, can be derived by interpolation in the multidimensional space defined by adverbs. The dimension of this space is defined by the number of adverbs. One adverb axis describes one emotional (happy or sad) or kinematic (uphill or downhill) characteristic of the motion. The interpolation is applied individually to each B-splines control point for each DOF and at each keytime. And each interpolation is formulated as a combination of linear polynomials and radial basis functions. "The polynomial function provides an overall approximation to the space defined by example motions. The radial bases locally adjusted the polynomial to interpolate the examples motions themselves" [18]. Thus, given a set of required adverb values $\mathbf{p}$, the interpolated $k^{th}$ control point in the $j^{th}$ DOF function of the resulting motion is defined in Equation 2.7.

$$b_{jk}(\mathbf{p}) = \sum_{i=1}^{NumExamples} r_{ijk} R_i(\mathbf{p}) + \sum_{l=0}^{NumAdverbs} a_{jkl} A_l(\mathbf{p}) \tag{2.7}$$

where $r_{ijk}$ and $R_i$ are, respectively, the weight of the radial basis function and the radial basis function. $a_{jkl}$ and $A_l$ are, respectively, the coefficient and the basis function. The $m^{th}$ interpolated keytime is similarly defined in Equation 2.8.

$$K_m(\mathbf{p}) = \sum_{i=1}^{NumExamples} r_{im} R_i(\mathbf{p}) + \sum_{l=0}^{NumAdverbs} a_{lm} A_l(\mathbf{p}) \tag{2.8}$$

There are $NumCP \times NumDOF$ control point interpolations (Equation 2.7) and $NumKeyTimes$ keytime interpolations (Equation 2.8) to generate a new motion. In

[18], the authors demonstrate how to evaluate the coefficients and the basis functions in 2.7 and 2.8.

Verbs and adverbs mechanism is similar to the work of D. J. Wiley et al. [22] discussed in the above section. Both techniques make use of interpolation in a multidimensional parameter space to generate a new motion. Verbs and adverbs mechanism uses a combination of radial basis functions and linear polynomials to approximate an in-between motion. And it requires $\mathbf{O}(n)$ motion examples to construct the parameter space and $\mathbf{O}(n^3)$ to compute the resulting motion. The work of D. J. Wiley directly applies linear interpolation and requires $\mathbf{O}(2^n)$ motion examples and $\mathbf{O}(2^n)$ to compute the resulting motion. $n$ is the number of parameters. Thus, verbs and adverbs mechanism is a more general method and scales better with the dimensionality of the parameter space. The additional difference is the different synchronization methods they apply before interpolation. The first one is based on key events, while the second one on a cycle-by-cycle basis. Note that a cycle also can be thought of as a coarse-level event.

The interpolation synthesis discussed above is performed in the time domain. In the following subsection, we discuss interpolation performed in the frequency domain. Most of them are adapted from standard signal processing methods.

### 2.3.3 Interpolation by Multi-Resolution Analysis

A. Bruderlin et al. in [4] propose to generate a new motion by blending the corresponding frequency bands at different resolution levels of different motion data. This approach is based on the following intuition: the low-level frequency bands control the general motion patterns, while the higher-level frequency bands contain motion details, such as mode, style and personality. Thus, the change of coarse level affects the main pattern of the motion, while the change of finer level affects the style of motion. In this technique, motion data are decomposed into different lowpass and bandpass levels by a multi-resolution filter. Then during blending, different interpolation coefficient is used at different frequency levels independently. As a result, this may generate a rich variation of motion with a wide range of styles. After blending, the resulting motion is reconstructed by adding up all the blended

bandpass sequences plus the DC (direct current) term.

Motion decomposition is done by successively convolving the motion data with a B-spline kernel filter with a width of 5, while at each iteration, the motion data is down-sampled by a factor of two. In this way, the number of frequency bands is determined by the number of frames of a decomposed motion and is computed in Equation 2.9.

$$let \quad 2^n \leq m \leq 2^{n+1}, \quad then \quad fb = n \tag{2.9}$$

where $fb$ is the number of frequency bands. $m$ is the number of frames or motion length. Before decomposing, all the blended motion data need to be synchronized by re-parameterizing with a generic time (time-warping). Thus, the blended motions have the same length $m$ and the same frequency level $fb$ after decomposing.

In the framework of [4], the dynamic time warping algorithm is applied to automatically synchronize the blended motions. This involves two steps. First, the optimal correspondence frames are automatically detected based on different similarity analysis (discussed in the last part of this chapter). A grid is constructed to represent the frame correspondence. In the grid, rows and columns, respectively, represent the frames of two blended motion clips. Each cell records the distance measure between the two corresponding frames. An optimal path, which minimizes the sum of the distance, is computed using dynamic programming. The second step is to warp the second motion clip with the first clip based on the optimal path. It is done by a non-uniform mapping and includes three operations: substitution, deletion and insertion. In the optimal path (shown in Figure 2.6), a diagonal line indicates that one frame in clip B corresponds to one frame in clip A; a horizontal line means that multiple frames in clip B correspond to one frame in clip A; a vertical line means that one frame in clip B corresponds to multiple frames in clip A. During warping, for the first case, the frame in clip B is kept. In the second case, the mean of multiple frames in clip B is used to replace the corresponding multiple frames of the mean in clip B. In the last case, a cubic B-spline is used to interpolate the frame and its neighbor to get the required inserted multiple frames. After warping, clip B has the same length as clip A.

The details of the blending algorithm based on multi-resolution filtering are described in the following. Step 1 to 5 are performed simultaneously on all the

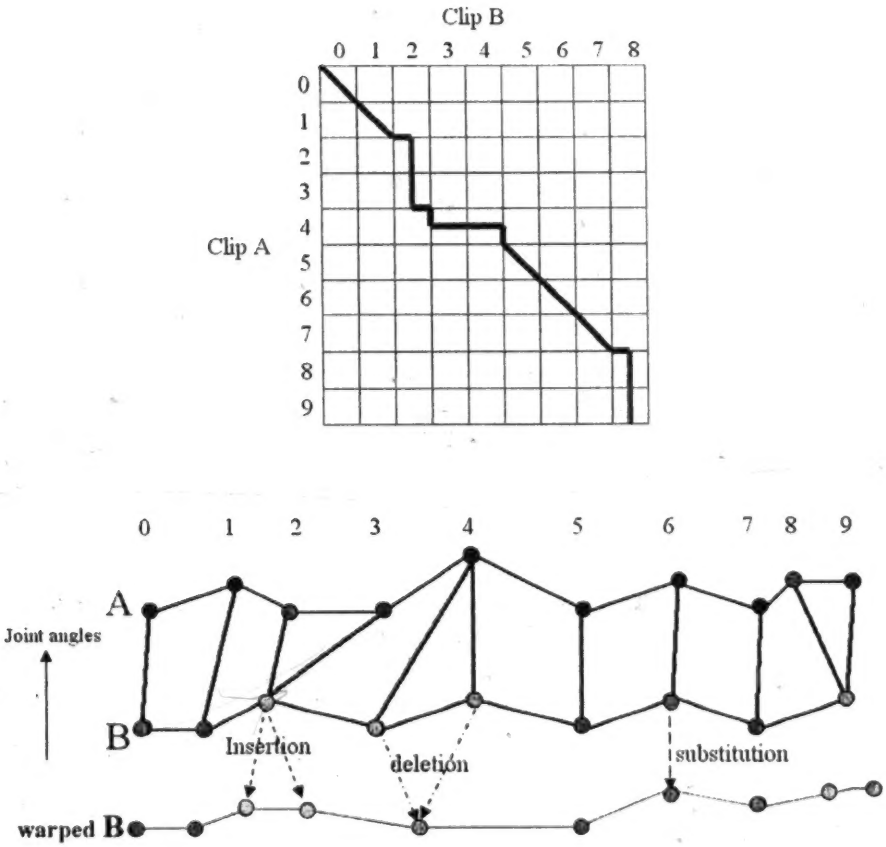



Figure 2.6: Top: the optimal path computed by dynamic programming. Bottom: the warping process based on the optimal path.

degrees of freedom for each motion dataset.

1. Calculate all the lowpass sequences by successively convolving the original motion data with a B-spline filter kernel. While at each iteration, the motion data is down-sampled by a factor of 2.

$$G_{k+1} = w \times G_k; \quad 0 \leq k < fb$$

where $G_0$ is the original motion sequence, $G_{fb}$ is the $DC$ term, $w$ is the B-spline filter kernel. At each iteration, the motion sequence is kept at the same length. The down-sampling is achieved by expanding the filter kernel at each level while zeros are inserted between the elements of the kernel. For example, the following shows the first three kernels.

$$w_1 = [\ 1/16 \quad 1/4 \quad 3/8 \quad 1/4 \quad 1/16\ ]$$

$$w_2 = [\ 1/16 \quad 0 \quad 1/4 \quad 0 \quad 3/8 \quad 0 \quad 1/4 \quad 0 \quad 1/16\ ]$$

$$w_3 = [\ 1/16 \quad 0 \quad 0 \quad 0 \quad 1/4 \quad 0 \quad 0 \quad 0 \quad 3/8 \quad 0 \quad 0 \quad 0 \quad 1/4 \quad 0 \quad 0 \quad 0 \quad 1/16\ ]$$

2. Calculate all the bandpass sequences by repeatedly subtracting two successive lowpass sequences. As all the lowpass sequences have the same length, the lowpass band at a lower level ($G_{k+1}$) does not require to be expanded first.

$$L_k = G_k - G_{k+1} \quad 0 \leq k < fb$$

3. Adjust the amplitude of each band and multiply $L_k$ by their modified amplitudes. This step increases the variations of the resulting motions. For example, increasing the high frequency band can add a nervous twitch to the movement, whereas increasing the lower frequency band can generate an attenuated, constrained action with reduced details.

4. Interpolate different bands at different levels. Each resolution level has an independent interpolation factor.

5. Reconstruct the resulting motion by adding up all the bandpass with the DC term.

$$G_0 = G_{fb} + \sum_{k=0}^{fb-1} L_k.$$

Another similar approach is the work described in [1], which incorporates wavelet analysis with motion blending to generate a new motion. Similarly, the original motion data are decomposed into multi-resolution levels by applying discrete wavelet transformation and normal blending is performed to each resolution level independently. Then the inverse discrete wavelet transform is applied to the resulting blended motion to reconstruct the final result from different resolution levels.

### 2.3.4 Interpolation by Fourier Expansion

In the work of [20], human locomotion with new emotions is synthesized by interpolation on the basis of Fourier expansions of the original motion data. For the $m^{th}$ joint, its orientation is represented as a Fourier expansion series shown in Equation 2.10.

$$\Theta_m(t) = A_{m0} + \sum_{n \geq 1} A_{mn} \sin(nt + \Phi_{mn}) \tag{2.10}$$

Here, for simplicity, one $\Theta_m(t)$ denotes 3 DOF for each axis, that is, $\Theta_x^m(t), \Theta_y^m(t)$ or $\Theta_z^m(t)$. As this technique is applied to periodic motion only, such as walking and running, the period of $\Theta_m(t)$ for each joint in one motion data is the same value $T_\Theta$. And the Fourier coefficients can be estimated from the discrete sample values $\Theta_m(t_p)$, $t_p \in [-T_\Theta/2, T_\Theta/2]$. After re-scaling the time parameter $t$ (called time re-parametrization), the period of the Fourier expansion series is normalized to $2\pi$. In this way, the blended motion data are synchronized. Similarly, the Fourier representation of another motion data is defined in Equation 2.11.

$$\Pi_m(t) = B_{m0} + \sum_{n \geq 1} B_{mn} \sin(nt + \Psi_{mn}) \tag{2.11}$$

The re-scaled Fourier representation of the original motion data can be used to generate new motions with different styles. The interpolation between two sets of motion data are defined in Equation 2.12.

$$\Xi_m(s,t) = (1-s)A_{m0} + sB_{m0} + \sum_{n\geq 1}((1-s)A_{mn} + sB_{mn})\sin(nt + (1-s)\Phi_{mn} + s\Psi_{mn}) \quad (2.12)$$

where $s$ is the interpolation coefficient, and $s \in [0,1]$. If $s$ varies from 0 to 1, then the interpolation result $\Xi_m$ continuously varies from $\Theta_m$ to $\Pi_m$. In this case, $s$ is a parameter defined in the frequency domain. Another interpolation form based on the Fourier representation is defined in Equation 2.13, where $s$ is a parameter in the time domain. Both forms can generate similar variations of the original motions by using different interpolation coefficients.

$$\Xi_m(s,t) = (1-s)\Theta_m(t) + s\Pi_m(t). \quad (2.13)$$

In addition, the authors also demonstrate how to use the Fourier representation to control some high-level motion characteristics, which involve step length, speed, hip position, etc. For example, in Equation 2.10, replace $A_{mn}$ with $stepA_{mn}$, thus, the step length can be adjusted by modifying the spectrum component in its Fourier representation. An alternative way to generate motion with new characteristics is also illustrated. Actually the idea is very simple and based on the interpolation techniques. Motion characteristics (called emotion component in the original paper) are extracted by taking the difference of the Fourier representations of two motion data, then the extracted characteristic is applied to the third motion. A new motion with the required characteristics is generated. For example, the method gets "briskness" from a normal walk and a brisk walk, Then a brisk run is generated by adding the "briskness" to the Fourier expansion series of a normal run.

The major limitation of this method is that it only addresses similar periodic human motions, like walking and running. Thus, motion synchronization is easily dealt with by uniform scaling.

### 2.3.5 Summary of Interpolation Synthesis

Motion interpolation is a purely geometric technique, and cannot guarantee to produce physically possible motions. In general, the interpolation of two dissimilar motion clips may generate distorted and unnatural results. Because the input data

implicitly contain the kinematic and dynamic constraints and if the two clips are similar enough, the result is also similar to the source, and most likely satisfies the constraints and looks realistic. This is the reason why most interpolation synthesis techniques are performed over a set of similar motion examples.

Additionally, the interpolated data must first be aligned in the time domain. In other words, interpolated motions need to be re-parameterized with the generic time. Thus, at a given generic time, all the motions and their interpolation results have similar structure elements. The similarity between interpolated motion data is further improved by motion synchronization. Now there are two approaches to address this problem. In some interpolation techniques like the one in [4], non-uniform time scaling is applied. Others [18, 20, 22] use a uniform time scaling approach.

## 2.4 Reassembling Synthesis

Inspired by the research work of video texture, a number of reassembling-based motion synthesis algorithms are proposed to create a new motion sequence by directly stitching different motion pieces together [2, 12, 13]. This category of motion synthesis is based on the fact that possible transitions can be generated between different motion clips when appropriate transition points are carefully selected. After cutting and pasting, different source motion pieces are reordered to compose a new sequence, while at the same time, the smooth and natural transitions between them are generated. Thus, a richer variety of motion sequence can be created from a given motion capture dataset.

Meanwhile, instead of directly using the original motion data, some researchers use a statistical model to simplify human motions for motion synthesis [3, 15]. As the statistical approach uses a generalization of motion, they cannot preserve the subtle details of the original motion. Moreover, the statistical approach does not involve motion similarity analysis. In the following subsections, we only focus on the reassembling synthesis approaches that directly use motion capture data and apply motion similarity analysis.

Most motion reassembling schemes generally include three steps: detecting transitions, constructing motion graphs, and searching the graph for an optimal path that determines the required motion sequence satisfying the user specifications. The motion transition points are detected based on different motion similarity analysis. The quality of the resulting motion sequence directly depends on the similarity analysis results. And typically, similarity analysis requires $\mathbf{O}(n^2)$ computation time, where $n$ is the number of frames in the dataset. Therefore motion similarity analysis is a very important but a time-consuming step. The second step is to build graph structures based on the detected possible transitions between different clips. Three types of graph structures exist in the current research area, namely, one-level graph, two-level graph and hierarchical graph. Generally, the graph encodes the motion data and their possible transitions. A path or a walk in the graph infers a possible motion sequence. A random path can create a continuous motion sequence that is regularly applied to generate a crowd animation or screen saver programs. However, in most situations, different user interface techniques are used to control the synthesized motion. According to user specifications, an objective function is defined to evaluate possible paths. Different searching strategies are applied to efficiently find an optimal path in the graph.

So in the following subsections, three typical motion reassembling schemes are introduced. First we introduce the graph structure, then discuss how to build it, and finally how to use motion graphs to generate a new motion sequence. Due to the importance of motion similarity analysis, it is discussed at the end of this chapter. The work of L. Kovar and his colleagues [12] generates a one-level motion graph. Then according to the path the user specifies, the combination of incremental search and the branch and bound search is performed to synthesize a new motion sequence. J. Lee [13] proposes to construct a two-level motion graph. Three interface techniques are used to control the search process, in particular, the user selects from a list of available choices, sketches a path and acts out an action in front of a video camera. In the video-based interface, the best fit of the performed action is found in the graph. O. Arikan and D. A. Forsyth [2] construct a hierarchy of graphs to represent the connectivity of a motion database and perform randomized search to find the motion that meets user specified constraints. In the following, the review

focuses on the graph structure, the searching strategies and the way in which the user controls the search process.

### 2.4.1 One-level Motion Graph

The one-level motion graph presented in [12] is a directed graph that represents how the captured clips can be re-assembled in a new order. Each edge corresponds to a motion clip. Some edges are original clips, while others are generated transitions. Each node represents the point at which the incoming clips can be seamlessly followed by the outgoing clips. Then a path or a walk in the graph composed of a sequence of edges suggests a possible new motion sequence. Figure 2.7 shows the structure of a simple motion graph and a possible motion sequence.

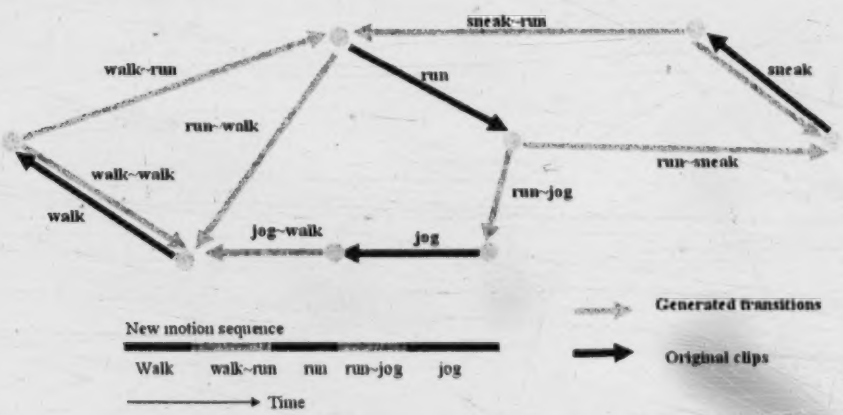



Figure 2.7: The structure of a one-level motion graph.

Building motion graphs includes the following steps. First, based on the results of motion similarity analysis, similar frame pairs with local minimal difference are selected as transition points; then the transitions between different clips are generated by linear blending. Finally, the problematic nodes are eliminated to obtain a more connective graph as described in the following. There are two kinds of problematic nodes to remove. One is called dead node, which does not belong to any cycle. Once such a node is entered, no more clips can be added to the synthesized motion. The other is called sink node. It belongs to only part of one or more cycles,

but is only able to reach a small fraction of the graph. So when a sink node is entered, the generated motion is confined to a small part of the graph (see Figure 2.8). The graph pruning is done as follows. Each frame is associated with a set of labels, which describe the motion patterns and possible constraints. The labels are manually annotated before the graph is built. For each label set, all the edges (motion clips and their transitions) whose frames with the exact label set, form a subgraph. Then the strongly connected components (SCC) are computed for each subgraph, where SCC is a maximal set of nodes such that there is a connected path between any two nodes in the SCC. Then edges that do not belong to the SCC and nodes with no edges attached are deleted. The pruned graph guarantees that an arbitrarily long motion sequence can be generated by traversing the graph.

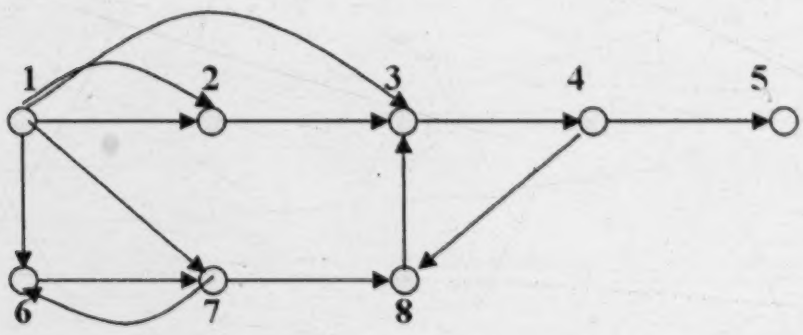



Figure 2.8: The problematic nodes in the graph. Node 5 is a dead node. Node 4 is a sink node.

The goal of constructing motion graphs is to generate a new motion stream that conforms to user specifications. This is achieved by searching for an optimal path that minimizes an objective function in the graph. The objective function is defined to evaluate how well a possible path meets the user requirements as shown in Equation 2.14.

$$f(w) = f([e_1, \ldots, e_n]) = \sum_{i=1}^{n} g([e_1, \ldots, e_{i-1}], e_i) \tag{2.14}$$

where path $w$ is composed of a sequence of edges $e_1, \ldots, e_n$; $f(w)$ is the cost of path $w$; $g(w, e)$ is a non-negative scalar error function, which describes the error when an

edge $e$ is added to an existing path $w$.

While searching, a halting condition is specified by the user to decide when to stop the search process. The search cost for a global optimal path is too expensive, even within a small dataset ( $20 \sim 200$ sec in [12] ). Thus, a branch and bound local search strategy is used to increase the search efficiency. The purpose of searching is to find a path that minimizes the error function $f$. As the error $f(w)$ of path $w$ is the lower bound on $f(w+v)$ for any edge $v$, so only the current optimal path $w_{opt}$ is kept and the other branches whose error exceeds $fw_{opt}$ can be discarded immediately. Although this strategy can reduce the number of searches to some degree, but the searching is still exponential. So an incremental searching is used to tradeoff some optimality for the searching speed. At each step, the optimal path of $n$ frames is found by the branch and bound searching, then the first $m$ frames are kept, the $m+1^{th}$ frame is used as the starting point for another search. In their implementation, the value of $n$ varies from 80 to 120 (2.67 to 4 sec), while the value of $m$ varies from 25 to 30 (about 1 sec).

In [12], the authors also demonstrate how to adapt the general searching process discussed above to the problem for path synthesis. The projection of the root joint on the floor at each frame forms a piecewise planar curve, which is called a path. The distance between the synthesized path and the specified path is defined as the edge error function. And it is computed as in Equation 2.15.

$$g(w,e) = \sum_{i=1}^{n} \|P'(s(e_i)) - P(s(e_i))\|^2 \tag{2.15}$$

$P'(s)$ is the point on the synthesized path whose arc length from the starting point is $s$. $P(s)$ is similarly defined for the desired path. $s(e_i)$ is the arc length from frame 0 to frame $i$ for edge $e$. The halting condition for path synthesis is when the current path length equals or exceeds the length of the specified path.

This approach can create realistic and controllable motion sequence from a small dataset (the largest dataset involves thousands of frames and takes over 200 sec). Path synthesis is a simple and easy way to control the locations and orientations of the resulting motion sequence. Additionally, the motion pattern constraint (like walking or running) is satisfied by confining the search to a subgraph defined by an annotated label set. The major problem of the one-level graph approach is the scal-

ability. The edges of one node will quickly increase with the size of motion dataset, which will make the search for an optimal path very difficult, if not impossible.

### 2.4.2 Two-level Motion Graph

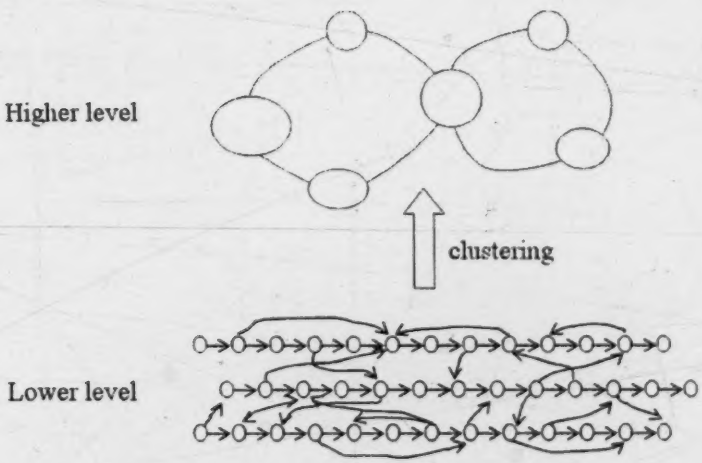



Figure 2.9: The structure of a two-level motion graph. The lower level encodes source motion data and their possible transitions on a frame-by-frame basis. The higher level records the clusters and their connections.

In [13], a two-level graph is constructed as shown in Figure 2.9. The lower layer keeps the details of the original motion data and their possible transitions on a frame-by-frame basis. The higher layer is the generalization of the lower layer by grouping the frames with similar poses into clusters. At the lower layer, the motion data in the database is represented as a first order Markov process. The transition from one frame to another frame depends only on the similarity between these two frames. The transition probability is computed as in Equation 2.16.

$$P_{i,j} \propto \exp(-D_{i,j}/\delta) \tag{2.16}$$

where $D_{i,j}$ is the similarity metric between frame $i$ and frame $j$. $\delta$ controls the mapping from the similarity measure to the transition probability. Smaller values

of $\delta$ emphasize very smooth transitions, while the larger values of $\delta$ trade some transition smoothness for a wider variation of the resulting motion.

If the tested motion database has $n$ frames, then the number of possible transitions becomes $\mathbf{O}(n^2)$. So the lower layer graph needs pruning. The following four pruning rules are used to reduce the number of transitions.

- Contact constraint. Transitions should happen between two similar frames only when they have similar contact states. It is because the contact state is a very important constraint for the motion. For example, a transition is not allowed from a frame when the foot is leaving the floor to another frame when the foot is touching the floor. For a motion clip, the system detects the contact state by considering the relative velocity and proximity between the characters and other objects in the same virtual environment.

- Similarity constraint. Only the transitions between similar frame pairs are kept. That is when the similarity measure between two frames is below a user specified threshold, the transition probability is set to zero.

- Local maximum constraint. Among the similar transitions, only the best transition with the maximum probability is retained.

- Connectedness constraint. Dead ends or sink nodes are discarded. This is similar to the pruning process in the one-level motion graph.

Even after pruning, the lower layer representation is too complicated to efficiently determine the required motion. So it needs to be further simplified. The higher layer is built based on clustering analysis. Each cluster is a group of similar frames which have similar poses. Then for each frame, all the possible clusters that may be transitioned to from this frame are found. These clusters are organized as a tree structure, called the cluster tree. In this case, clusters capture the similarities between frames, while the connections between different frames are captured by the cluster trees. The process to build a cluster tree for frame $i$ is described as follows (see Figure 2.10):

- the cluster to which frame $i$ belongs becomes the root of the tree

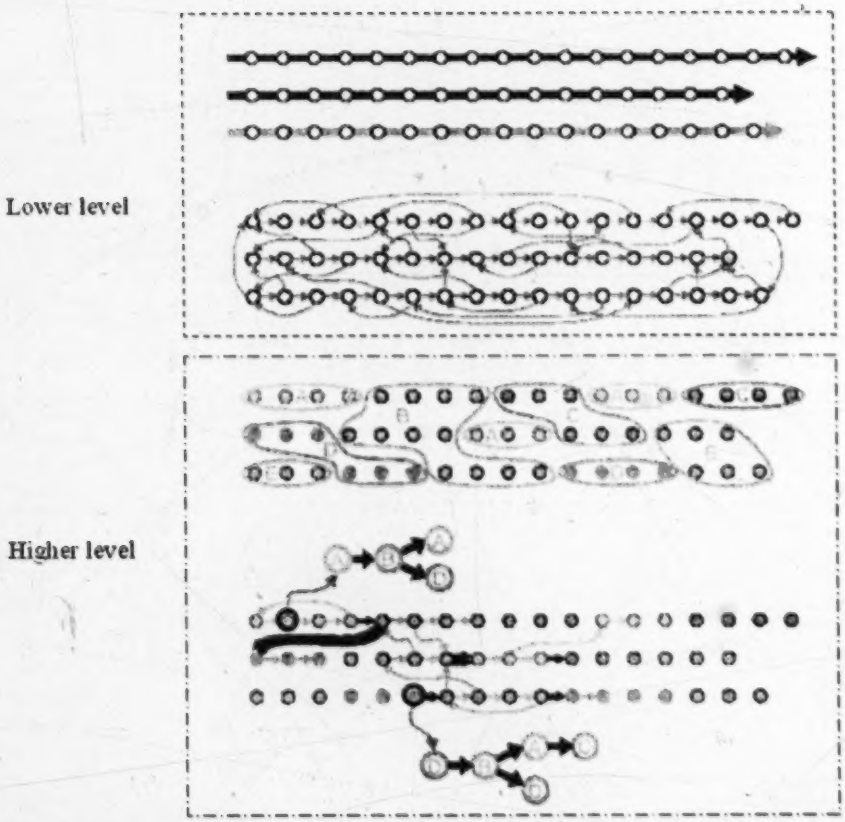



Figure 2.10: The process for building cluster trees. First row: Original motion data on a frame-by-frame basis. Each node is a frame, and each edge between two nodes means that one frame can be seamlessly followed by the second frame. Second row: Transitions denoted by arrows in the lower layer are generated. Third row: Similar frames are grouped into clusters. Last row: Two cluster trees are formed for the 2 frames marked with red circles. (This figure is taken from the original paper.)

- find the transition with the maximum transition probability to another cluster
- if the number of frames required to reach that cluster is within a specified threshold, add the cluster as a child to the current node
- repeat step 2 and 3 until no more clusters are found.

Each frame in the lower layer is associated with a cluster tree. The cluster tree encodes the valid transitions for this frame. The cluster trees for all the frames in the motion database form the higher level graph, also called the cluster forest. Thus, the two-level graph structures are closely linked by cluster trees.

This two-level graph structure not only makes efficient search possible, but also provides the presentation in the user interface. For example, in the case of choice-based interface, a list of options is provided to the user to select according to the cluster tree of the current frame (state). Here the concept of a cluster path is introduced. A cluster path is a path from the root (current frame) of a cluster tree to one of its leaves. If a cluster tree has $k$ leaves, then it has $k$ cluster paths. Each cluster path defines a set of possible motion sequences that can follow the current frame. For each set of possible motion sequences available to the current frame, the one with the highest probability, also called the most probable motion, is found as follows. Within one cluster path, all the possible paths that start from the root node are searched and evaluated by computing the sum of probabilities along the path. The path with the maximum probability is the most probable path. Each cluster path has one most probable path. Then according to all such paths, a set of possible action choices next to the current frame is provided to the user to choose. Typically, the available action choices at the current frame are represented as example poses, paths, or footprints.

The researchers in [13] also demonstrate two other interface techniques to control the graph search. One is path sketches, and the other is video performance. In the path sketches interface, the user specifies a desired path and the graph is searched to find a motion sequence that follows the path. Two ways are implemented to find the resulting motion. One is to select the most appropriate action available at the current time based on cluster paths. The distance between the desired path and the synthesized motion defined by the most probable path within each cluster

path is evaluated. The path with the minimum distance is selected. In the second method, clustering is not involved and searching is directly performed through the lower graph, which is similar to the path synthesis in the one-level motion graph. The distance between the desired path and the actual path is computed and the path with the minimum distance forms the resulting motion.

In the case of vision-based interface, a user acts out the desired motion in front of a camera, then the best fit of the performed motion in the graph is found. The difference between the performed action and the synthesized motion is defined as the path cost to evaluate each possible path. Visual features based on silhouette are extracted from the video and are used to compute the distance between a possible action and the performed action. In practice, as the direction of the user with respect to the camera is unknown, the original motions in the dataset need to be represented with different viewpoints. 18 different camera viewpoints are used in [13].

In this work, the authors mainly focus on how to use the three types of user interfaces to control the synthesized motion sequence. The vision-based interface provides a complete control over the synthesized motion due to the high dimensional input, but it is not a general interface as special devices are needed. The choice-based interface is most helpful to provide the user with the information what motions are available to the current state in the dataset. And the sketch-based interface is a simpler and more natural way to control the directions and positions of the animated character. Though the two-level graph structure is used to increase the search efficiency, its scalability with large databases is a potential problem.

### 2.4.3 Hierarchical Motion Graph

In the hierarchical motion graph approach, the motion database is represented as a hierarchical structure, defined as $G_1 \leftarrow G_2 \leftarrow G_3 \leftarrow \ldots \leftarrow G_n \leftarrow G$, where $G_1$ is the coarsest level graph, $G$ is the finest level. Random search is performed at a variety of levels to generate an optimal motion sequence which meets the user specified constraints.

The lowest level $G$ is also a directed graph, which captures the details of the

original motion data and their possible transitions. But its structure is different from that of [12]. At this level, a node represents an individual motion clip, and an edge between two frames represents a possible transition between the two frames. Here, $s_i$ denotes the $i^{th}$ frame in clip $s$, while $t_j$ denotes the $j^{th}$ in clip $t$. Then an edge from $s_i$ to $t_j$ means that frame $s_i$ can be smoothly followed by frame $t_j$. A cost value is also associated with an edge that describes the transition probability between the frame pair. The cost of an edge is computed based on a similarity measure discussed in Section 2.6. Therefore an edge $e$ from $s_i$ to $t_j$ is labeled by the following 5 terms:

- $fromClip(e) = s$
- $toClip(e) = t$
- $fromFrame(e) = i$
- $toFrame(e) = j$
- $cost(e) = D_{i,j}$

where $D_{i,j}$ is the similarity measure between the two frames $i$ and $j$. In this setting, any sequence of edges $e_1, e_2, \ldots, e_n$, defines a valid motion sequence, when the following two conditions are satisfied.

$$toClip(e_i) = fromClip(e_{i+1})$$

$$toFrame(e_i) < fromFrame(e_{i+1})$$

A valid path involving 6 edges and the resulting motion sequence are illustrated in Figure 2.11.

From similarity analysis, it is known that there are many possible transitions between any two motion sequences. Thus, the finest level graph involving a huge amount of edges is too complicated to search for an optimal path within a reasonable time. To increase the search efficiency, the finest level graph is summarized and the coarsest level graph $G_1$ is used to help in the search. The highest level graph $G_1$ is built by clustering all the edges of $G$. The clustering process is described as follows. All the edges between two nodes $s$ and $t$ can be represented in a matrix $P_{st}$, where

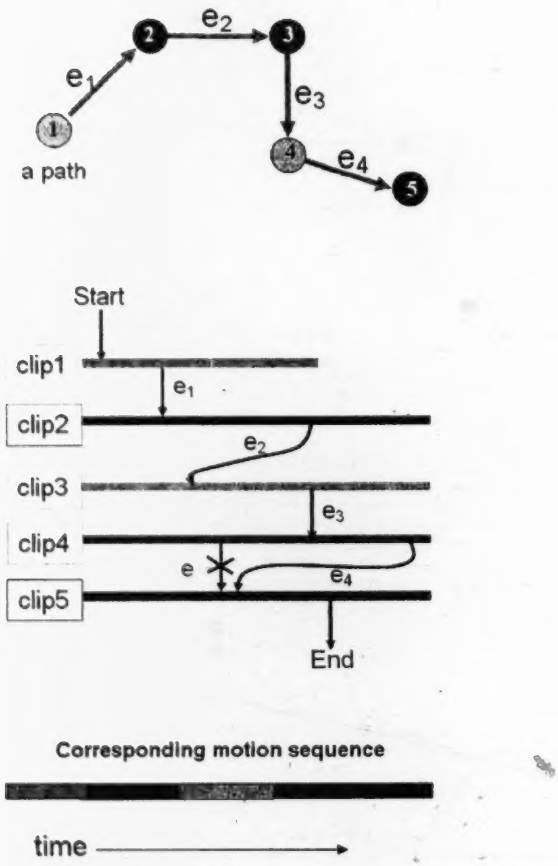



Figure 2.11: A valid path and its corresponding motion sequence. The path involves 6 edges and 4 clips. The first edge and the last edge are, respectively, used to mark the starting and end point of the path. If $e_4$ is replaced with $e$, the path becomes invalid because $fromFrame(e)$ appears before $toFrame(e_3)$ in time, the second condition is not satisfied.

the $ij^{th}$ entry is the cost value of the edge connecting the frame pair $s_i$ and $t_j$. If no such edge exists, then infinity is assigned (Equation 2.17).

$$(P_{st})_{ij} = \begin{cases} C(s_i, t_j) & \text{if the similarity measure} > \text{threshold;} \\ \infty & \text{otherwise.} \end{cases} \tag{2.17}$$

If two frames are similar, then the frames in their neighborhood most likely are similar too. Hence, the resulting similar frame pairs after similarity analysis between two nodes will form several small groups (shown in Figure 2.12). In the matrix $P_{st}$, columns correspond to the $fromFrame$ term, rows correspond to the $toFrame$ term. In this approach, the clustering between two nodes are found based on the two terms, $fromFrame$ and $toFrame$, instead of based on frame poses that are used in the two-level graph approach. For simplification, clustering is performed over any two nodes separately.

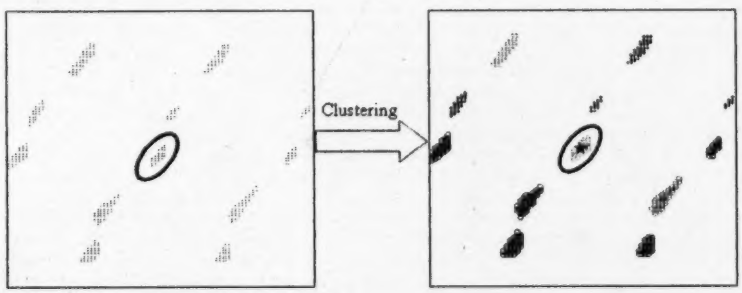



Figure 2.12: The clustering process for building top level graph $G_1$. Every edge between two nodes is represented as an entry in a matrix. Similar frame pairs in a neighborhood form a cluster (marked with a blue circle). The center (marked with a red star) of these edges becomes one of the edges of $G_1$

After clustering, the nodes in $G_1$ are kept the same as $G$, while the edges in $G_1$ represent the clusters of edges in $G$. Hence, one edge in the top level graph $G_1$ represents many edges in the lowest level graph $G$. Then each edge in $G_1$ is split into two parts to generate a lower level $G_2$ by the k-means clustering method with $k = 2$. In a similar way, the intermediate levels are generated until the lowest level edge of $G$ is reached. Thus, at the highest level graph $G_1$, each edge is the root of a binary tree and represents all the edges in a close neighborhood. The leaf edges represent possible transitions of the original motion data and the intermediate edges represent the averages of all the leaf edges beneath them (Figure 2.13).

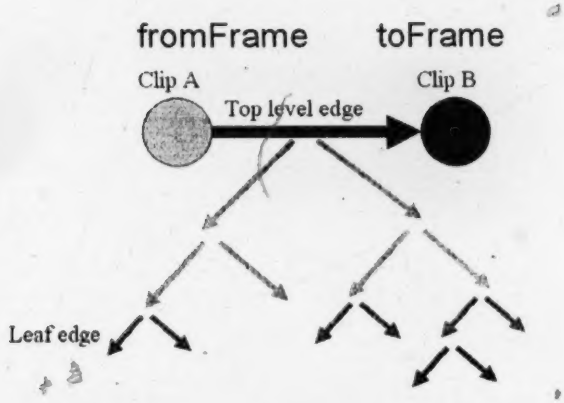



Figure 2.13: An edge at the top level graph is represented by a binary tree.

This hierarchical structure makes an efficient search possible. Before searching, the user specifies two types of constraints that the resulting motion sequences need to satisfy. A hard constraint means that it must be satisfied exactly. For example, the character is required to have a specific pose (like a ballet dancing pose) at a particular time. This kind of hard constraints will restrict the search to within a particular part (some nodes) of the graph. A soft constraint means that it can not be met exactly. Some soft constraint examples are listed as follows. Soft constraints provide an objective function that reflects how well soft constraints are satisfied and how close a valid path is to the required one.

- The total number of frames for the resulting motion should be a particular number.
- The motion should not penetrate any objects in the environment.
- The body should be at a particular position and orientation at a particular time.
- A particular joint should be at a particular position at a particular time.
- The motion should have a specified style (such as happy or energetic) at a specific time.

During graph searching, a random search strategy is performed to find an optimal path which best satisfies the specified constraints. The search process is as follows.

1. Start with some random seed paths at the top level $G_1$.
2. Evaluate all the current paths and their possible mutations. The two types of mutations are described later.
3. Keep the current optimal path.
4. Add new seed paths.
5. Repeat step 2 to 4, until no better path is found or the user is satisfied with the current optimal path.

All possible motion sequences indicated by the valid paths are scored according to how well they meet the soft constraints. For each soft constraint, the difference between the actual value and the constraint is defined as the constraint cost. A path cost is the sum of costs of all the imposed soft constraints plus the sum of costs of all the edges along the path in Equation 2.18. The edge cost term controls how smooth the transitions between motion pieces are, or the smoothness of the resulting motions. Note that the hard constraints are always satisfied by restricting the searching to some particular nodes.

$$s(e_1 \dots e_n) = w_c \times \sum_{i=1}^{n} cost(e_i) + w_f \times F + w_b \times B + w_j \times J \qquad (2.18)$$

where $w_c, w_f, w_b$ and $w_j$ are, respectively, the weights for the smoothness of motion, the length of motion constraint, the body constraints involving position and orientation, and the joint constraints. $F$ is the difference between the specified number of frames and the actual number of frames in the path. $B$ is the distance between the position and orientation of the constraints versus the actual position and orientation of the body. $J$ is similarly defined to the joint constraint.

During searching, two types of path mutations are considered.

1. Delete the portion of the path between any two edges and replace it with one or two edges at the top level. This will introduce a better motion sequence

(the path with a lower cost is kept), and the computation time is $\mathbf{O}(n^2)$, where $n$ is the average of edge number going out from a node at the top level graph.

2. Replace every two edges with their children. This mutation will push the searching down to a finer level and find the substantial difference among a group of edges.

In this framework, the motion synthesis is modeled as an under-constrained problem. For many types of constraints, different motion sequences can be found to meet the soft constraints. The under-constrained nature motivates that the random search strategy and two possible mutations can be applied. So after the random search, a possible solution is found, but it is not guaranteed that the generated motion is the one user wants. Since the synthesis is an interactive process, at anytime, users can check the current resulting motion, and then adjust or add more constraints to get their required motion sequence. Hence, the flexibility of the constraint types a user can specified is one of its advantages. The appropriate constraint set complied with the original database plays an important role to the success of this technique. For example, the specified constraint set may involve only one constraint: catching a ball at frame 100. If no source motion includes this type of motion, the specified constraint will not be satisfied. The other advantage is the graph's hierarchical structure that makes the interactive search ($\mathbf{O}(n^2)$) possible, where $n$ is the average of edge number going out from a node at the top level graph.

### 2.4.4 Summary of Reassembling Synthesis

Reassembling-based synthesis techniques are able to create continuously varied motion sequences based on a set of motion captured data. The rich variety of the resulting motion sequence is achieved by identifying possible transitions between different motion clips. Therefore, the transition points selection is a critical part and directly affects the quality of the resulting motion sequence.

Since the synthesized motions are strictly bounded to the original motion data, the size and quality of the dataset also become key to the success of this approach. For example, if the motion source does not contain any "turning left" motions,

the reassembling based synthesis will not be able to create motions that involves "turning left."

The main problem related with this approach is the scalability of algorithms. Different motion graph structures motivate different efficient searching algorithms. In the one-level graph, the incremental search and the bound and branch search are used. For the other two graph approaches, an efficient search is achieved by clustering. In the two-level graph, a cluster is a group of similar frames, then a cluster tree is built to encode several groups of possible transitions to each frame. In the hierarchical graph, a group of similar transitions in terms of $fromFrame$ and $toFrame$ are directly clustered to become a top-level edge. The latter two frameworks scale better than the one-level graph scheme.

## 2.5 Motion Transition Mechanisms

Motion transition is an integral part in the approach to generate a motion sequence by reordering different motion segments. It is also an important application for motion interpolation. Based on motion similarity analysis, different transition mechanisms can be used to generate motion with different qualities. Three main transition schemes exist in the current research field of computer animation, namely, spacetime transition, motion stitching, and motion blending as described in the following. But first we discuss the issues related with candidate transition point selection.

### 2.5.1 Candidate Transition Point Selection

In order to create a smooth transition between two motion clips, the poses, velocities, and constraints should be matched at the transition point. In other words, the two frames together with their neighboring frames should be similar enough in terms of body configuration, velocities and constraints. In some situations, to create seamless transitions between two different motion clips is impossible. One practical example is the direct transition from a back flip to a dance. This is because the two motion clips are too different and direct transition between them is not allowed.

So a similarity threshold for candidate transitions should be specified to mark the lower bound similarity for a possible smooth transition. And different practical applications require different thresholds. For example, the transition between walking and running requires a higher threshold, as people are very familiar with these two movements. A lower similarity threshold maybe used in the transition between two less familiar motions, such as two different dancing clips.

### 2.5.2 Transition by Using Spacetime Constraints

In [19], the authors propose a complex semi-automatic algorithm to generate smooth and dynamically plausible transitions between two motion clips. During the transition period, the motion of the root, support limbs and non-support limbs are respectively dealt with. Here, a support limb is defined as the kinematic chain from the support point (e.g. a foot on the floor) back up the kinematic tree to the root. This involves the three processes described as follows.

The root motion is computed by linearly interpolating between the end of the first clip and the beginning of the second clip. The y-component of root translation is directly interpolated between the y-component at the beginning and the end of the transition. The x and z coordinates of the root position are linearly interpolated based on the integration of velocities (shown in Equation 2.19).

$$\mathbf{p}(t) = \mathbf{p}(t_1) + \int_{t_1}^{t} \left\{ \mathbf{v}_1 \left( 1 - \frac{\alpha - t_1}{t_2 - t_1} \right) + \mathbf{v}_2 \frac{\alpha - t_1}{t_2 - t_1} \right\} d\alpha \qquad (2.19)$$

where $\mathbf{v}_1$ and $\mathbf{v}_2$ are the root velocities in the $xz$ plane at time $t_1$ and $t_2$. $\mathbf{p}(t)$ and $\mathbf{p}(t_1)$ are the root positions in the $xz$ plane at time $t$ and $t_1$, respectively.

The motion of the support limbs is controlled by inverse kinematic constraints. Solving the inverse kinematic problem is an optimal procedure, which minimizes the deviation from the desired constraints. First the support points that serve as footprint constraints are detected by the system or by user's specifications. The system can automatically locate them by finding the frames in which the foot remains within a small bounding box over an extended period of time. Solving the inverse

kinematic constraints is an optimizing process which minimizes the difference between the actual positions and the desired positions of the support limbs over the constrained frames. For each constraint $k$, the deviation is defined in Equation 2.20.

$$r_k(t) = \|\mathbf{p}_k(t) - \tilde{\mathbf{p}}_k(t)\|^2 \tag{2.20}$$

where $\mathbf{p}_k$ is the actual position of the constraint frame and $\tilde{\mathbf{p}}_k(t)$ is the desired position. So the total error $R$ is the integral over the constrained time interval of the sum of all the $n$ constraints (see Equation 2.21).

$$R = \int_{t_1}^{t_2} \sum_{k=1}^{n} r_k(t)dt \tag{2.21}$$

The motions of non-support limbs, which do not support the body, are evaluated by the spacetime constraints optimization. Over the transition interval, joint torques are minimized to simulate metabolic energy minimization. So the objective function is formulated in Equation 2.22, where $\tau_i$ is the torque of the $i^{th}$ joint, $i$ is the joint index.

$$e = \min \int_{t_1}^{t_2} \sum_{t} \tau_i^2(t)dt \tag{2.22}$$

As this technique uses a combination of kinematics and dynamics, the results it generated can preserve kinematic constraints and basic dynamic properties. However, the disadvantage of this method is the high computational cost and its limited short transition period ($0.3s \sim 0.6s$). Typically, when the speed of two motion clips are quite different, the short transition period cannot satisfy the natural and realistic requirements. Additionally, the source motion segments are also assumed to be processed in advance. For example, the proper transition points have to be manually selected by the user.

### 2.5.3 Transition by Stitching

Motion stitching is a fast and easy way to generate a transition between two motion clips. This approach directly connects two clips at one frame while the discontinuity at the joined point is distributed in a small window around the joined point. When

the transition point is selected carefully, the discontinuity at the joined point is small, and after smoothing, seamless transition can be generated. For the cases when the difference between the two motion data is not small enough, the resulting motion typically have noticeable artifacts. In practice, a variety of smoothing methods are used to deal with the discontinuities at the connected point. One example is given as follows.

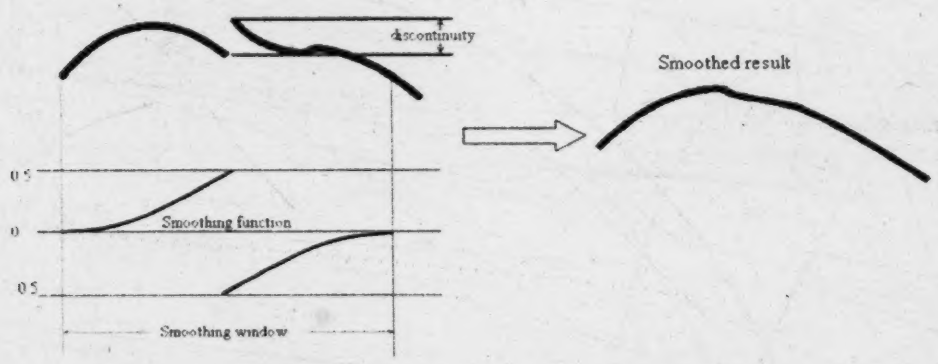



Figure 2.14: Transition by stitching: a quadratic smoothing function controls the amount of displacement of two connected clips in a smoothing window. Left top: The magnitude of discontinuity between two clips. Left bottom: The smoothing function. Right: The resulting transition after smoothing. (This figure is adopted from the original paper.)

In [2], a quadratic smoothing function is defined in Equation 2.23 and illustrated in Figure 2.14. The product between the magnitude of the discontinuity and the smoothing function determines the amount of displacement for every frame in the smoothing window. At the joined point, the end frame of the first clip has half the discontinuity, while the starting frame of the second clip has the left half. Thus, after smoothing, the discontinuity at the joined point goes away.

$$y(f) = \begin{cases} 0 & f < d - s \\ \frac{1}{2} \times (\frac{f-d+s}{s})^2 & d - s \leq f < d \\ -\frac{1}{2} \times (\frac{f-d+s}{s} - 2)^2 & d \leq f \leq d + s \\ 0 & f > d + s \end{cases} \tag{2.23}$$

where $d$ is the frame number at the joined point. $s$ is half the size of the smoothing window. $f$ is the frame index.

### 2.5.4 Transition by Interpolation

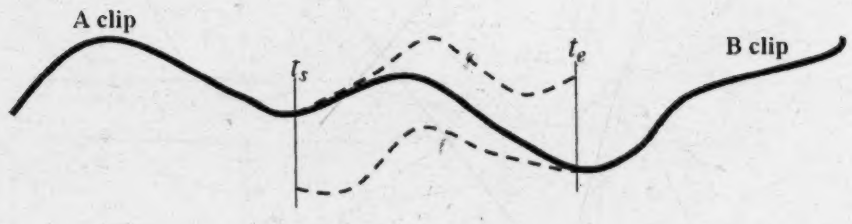



Figure 2.15: Transition by linear interpolation. $t_s$ and $t_e$ are the start transition point and end transition point, respectively.

Motion interpolation is one of the most important computer animation techniques, which "create a new motion by combining two or multiple different motion clips according to time-varying weights" [11]. In the case of motion transition, two motion clips are combined while the weight of the first clip changes from one to zero while the other varies inversely. In Figure 2.15, transition from clip A to clip B is achieved by linear interpolation. In some research works [4, 18], the blending is performed in the frequency domain, where the source motion data are decomposed into multi-resolution levels, then interpolation is performed in each level, the final motion is reconstructed by composing the interpolation at multiple levels. Generally, linear interpolation is performed to the root positions (Equation 2.24), while spherical linear interpolation is performed to the joint orientations (Equation 2.25).

$$\mathbf{p}_C = u \times \mathbf{p}_A + (1-u) \times \mathbf{p}_B \tag{2.24}$$

$$\mathbf{q}_C^i = \mathbf{q}_A^i \times \frac{\sin(uW)}{\sin(W)} + \mathbf{q}_B^i \times \frac{\sin((1-u)W)}{\sin(W)} \tag{2.25}$$
$$W = \cos^{-1}(\mathbf{q}_A^i \bullet \mathbf{q}_B^i)$$

where $\mathbf{p}_A$ and $\mathbf{p}_B$ are, respectively, the root positions of clip $A$ and $B$. $\mathbf{p}_C$ is the root position of the interpolated motion. $u$ is the interpolation coefficient. $\mathbf{q}_A^i$ is the quaternion that represents the orientation of the $i^{th}$ joint of clip $A$. $i$ is the joint

index. $\mathbf{q}_B^i$ and $\mathbf{q}_C^i$ are similarly defined to clip $B$ and the interpolated result. $W$ is the angle between the joint orientations of two source clips.

Motion transition based on the idea of motion interpolation has been applied to online or interactive generation of transition between two example motions, while the spacetime approach has been applied to off-line processing because of its expensive computational cost. Additionally, transitions by linear interpolation can be regarded as a special case of motion stitching, where discontinuities are linearly distributed over the transition region.

## 2.6 Motion Similarity Analysis

Motion similarity analysis is a critical stage in many motion editing techniques. In this section, two typical types of distance functions in recent research work are reviewed, and then the problems related to them are identified.

### 2.6.1 Similarity Based on Joint Attributes

Because motion capture data are usually directly represented by the joint angles, many research works prefer using similarity distance functions that are based on a weighted sum of multiple joint attributes, such as joint angles, joint velocities, root velocities and root accelerations. Two typical examples are given below.

J. Lee et al. in [13] present a similarity metric based on joint angles and joint velocities. The similarity between two frames is computed as follows.

$$D_{i,j} = w_p dp + w_a da + w_v dv \tag{2.26}$$

$$dp = \|\mathbf{p}_{i,0} - \mathbf{p}_{j,0}\|^2$$

$$da = \sum_{k=1}^{m} w_k \|log(\mathbf{q}_{j,k}^{-1}\mathbf{q}_{i,k})\|^2$$

$$dv = \sum_{k=1}^{m} w_k \|\mathbf{v}_{i,k} - \mathbf{v}_{j,k}\|^2$$

where $\mathbf{p}_{i,0}$ and $\mathbf{p}_{j,0}$ are the root joint (pelvis) translations at frame $i$ and frame $j$, respectively. $\mathbf{q}_{i,k}$ is the quaternion that represents the orientation of the $kth$ joint at frame $i$. $log(\mathbf{q}_{j,k}^{-1}\mathbf{q}_{i,k})$ is the angle by which the $kth$ joint rotates from the orientation at frame $i$ to the orientation at frame $j$. $\mathbf{v}_{i,k}$ is the velocity of the $kth$ joint at frame $i$. $\mathbf{q}_{j,k}$ and $\mathbf{v}_{j,k}$ are, respectively, the joint orientation and velocity at frame $j$. $k$ is the joint index, $m$ is the number of joints, and $w_k$ the joint weight to control the importance of the $kth$ joint. $w_p$, $w_a$ and $w_v$ are the attribute weights to accommodate for the relative importance of the different joint attributes.

The similarity function between frame $i$ and $j$ consists of three terms. The first one is the difference between root velocities. In the original paper [13], it is the difference between the global root translation with respect to the previous frame; in which case, it is the same as the difference in the root velocities. If we use the root positions instead of velocities, this method will become coordinates dependent. Then only the frames that are located nearby in the 3 dimensional space can be selected as the similar frame pairs. Hence, the similarity analysis results largely depend on the root positions. The other two terms are the difference of joint angles and the difference of joint velocities. The joint angle term captures the pose information. While the other two terms incorporate the kinematic information of the joints. From Equation 2.26, we can see that there are 3 weight parameters each for the 3 attributes. In the original work, no information is given on how to determine the values of these parameters. In this thesis, different attribute weights have been tried and it is found that the results can be quite different depending on the attribute weights (see Figure 2.16).

O. Arikan and D. A. Forsyth [2] define a similar distance function in Equation 2.27, except that it incorporates joint accelerations as well. This measure is based on joint positions and velocities in the root joint (pelvis) coordinates. There are four components in the equation, each of which represents a contribution to the similarity measure. $\sum_{k=1}^{m} w_k dp_k$ and $\sum_{k=1}^{m} w_k dv_k$ are, respectively, the difference in joint positions and joint velocities with respect to the root coordinates; $dv_0$ and $da_0$ are the difference of root velocities and accelerations. Each of the last two terms includes translation and orientation. $k$ is the joint index, $m$ the number of joints. $w_k$ is the joint weight to control the importance of the $kth$ joint; $w_p$, $w_v$, $w_{v0}$ and

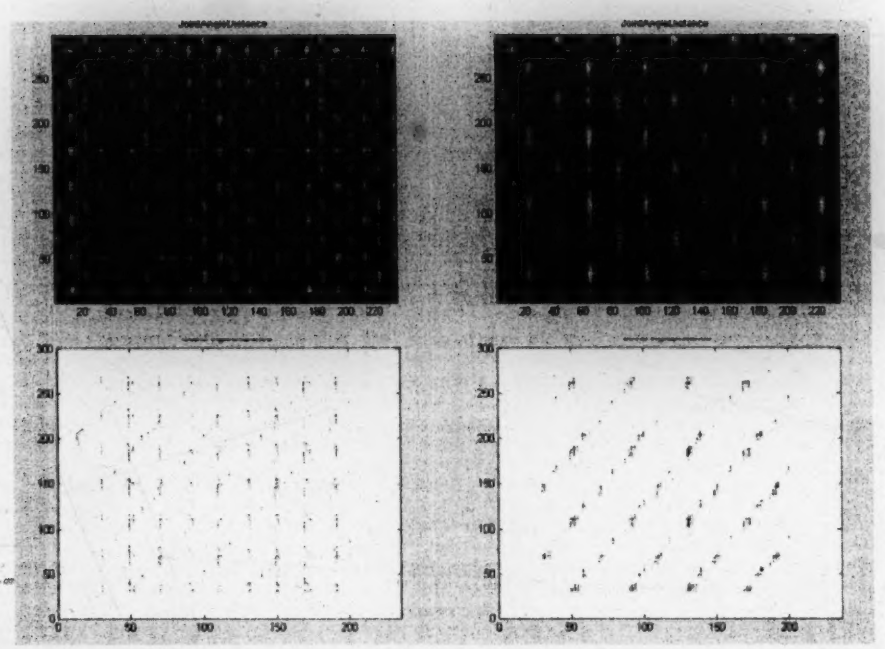


Figure 2.16: Similarity analysis based on joint angles with different attribute weight sets between left and right (columns correspond to frames from the first clip, rows correspond to frames from the second clip). Top: The distance matrix. Bottom: The resulting similar frame pairs are denoted by blue dots.

$w_{a0}$ are the attribute weights to control the relative importance of the different joint attributes. The difference of joint positions in the root frames defines the pose similarity. The other components incorporate kinematic information like velocities and accelerations.

$$D_{ij} = w_p \sum_{k=1}^{m} w_k dp_k + w_v \sum_{k=1}^{m} w_k dv_k + w_{v_0} dv_0 + w_{a_0} da_0 \tag{2.27}$$

The major drawback of this metric is on how to determine the optimal attribute weights. In the original work [2], the authors provide a clever way to handle the attribute weights. The maximum difference between sequential frames in the database is used to normalize each term. The normalization process could better balance the effects of the four features on similarity analysis. But the attribute weights after normalization might be dependent on different motion databases. In other words, the attribute weights are likely to change when the database includes motion streams with different special patterns. Figure 2.17 shows the different similarity analysis results generated for the same two clips in two different datasets.

### 2.6.2 Similarity Based on Point Clouds

In [12], two point sets driven by a skeleton are used to estimate the similarity between two frames. In fact, the point set of a frame can be formed by the joint positions at the frame and its neighboring frames, which form a window of the current frame. Each frame in the window has a different weight, called frame weight, which controls the relative importance of the frame with respect to the other frames in the window. Typically, the frame weight is tapered off towards the end of the window.

The similarity metric is defined as the minimum difference between the two point sets. In order to calculate the minimum difference in joint positions, an optimal 2D transformation matrix must be computed first. After transformation, the second point set can be aligned with the first point set. The similarity measure is defined in Equation 2.28.

$$D_{ij} = \min_{\theta, x_0, z_0} \sum_{f=1}^{F} w_f \sum_{k=1}^{m} w_k \| \mathbf{p}_{k,f}^{i} - \mathbf{T}_{\theta, x_0, z_0} \mathbf{p}_{k,f}^{j} \|^2 \tag{2.28}$$

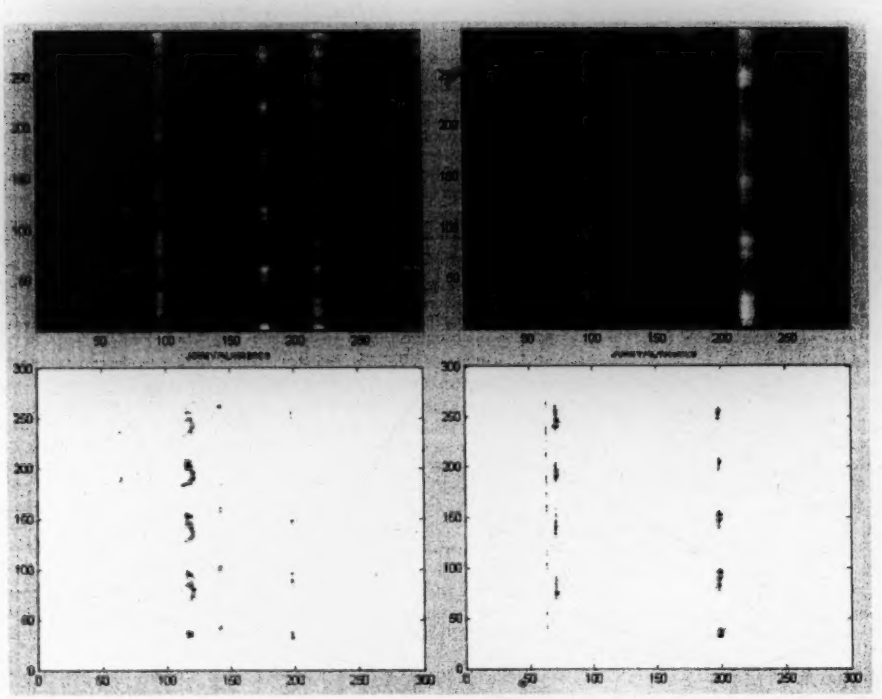


Figure 2.17: Similarity analysis based on joint velocities and accelerations in different datasets. Left: The dataset has only 2 clips. Right: The dataset has 25 clips. Top: The distance matrix. Bottom: The resulting similar frame pairs are denoted by blue dots.

where $\mathbf{p}^i_{k,f}$ is the $kth$ joint position in the $fth$ frame of the point set at frame $i$. $\mathbf{p}^j_{k,f}$ is similarly defined to the point set at frame $j$. $k$ is the joint index; $m$ is the number of joints; $w_k$ is the joint weight. $w_f$ is the frame weight; $f$ denotes the frame index within the window and $F$ is the number of frames in the window. $\mathbf{T}$ is the transformation matrix for rotating by an angle $\theta$ around the vertical axis, translating on the horizontal plane by $(x_0, z_0)$. In the original paper [12], a solution is given to find the transformation matrix $\mathbf{T}$ as in the following equations.

$$\theta = \arctan\left(\frac{\sum_i w_i\left(x_i z'_i - x'_i z_i\right) - \left(\bar{x}\bar{z}' - \bar{x}'\bar{z}\right)}{\sum_i w_i\left(x_i x'_i + z_i z'_i\right) - \left(\bar{x}\bar{x}' + \bar{z}\bar{z}'\right)}\right) \tag{2.29}$$

$$x_0 = \bar{x} - \bar{x}'_i \cos\theta - \bar{z}'_i \sin\theta \tag{2.30}$$

$$z_0 = \bar{z}_i + \bar{x}'_i \sin\theta - \bar{z}'_i \cos\theta \tag{2.31}$$

where $\bar{x} = \sum_i w_i x_i$ and the other barred terms are similarly defined.

In this approach, as joint positions are directly used, the similarity between body configurations of the frames can be captured. Also the consideration of a frame's neighborhood (8 frames in length) incorporates the kinematic differences between the two sequences.

The major disadvantage of this metric is that it uses a 2D transformation matrix as the optimal transformation matrix. The method assumes that all the motions happen at the same level as the ground. The optimal transformation matrix in [12] does not consider translation along the y-axis, and rotations around the x-axis or the z-axis. For example, when in one motion clip a character is running on a flat floor; and in the second clip, the character is walking on a ramp, the similarity result is shown in Figure 2.18. Obviously, when the vertical distance between the root joints of the two clips increases, their distance also increases. As a result, only the frames with root joints at the same level are selected as similar frame pairs. To solve this problem, one way is to compute the 3D transformation matrix by which the two point sets can be aligned. However, computing the 3D transformation matrix can be very time consuming.

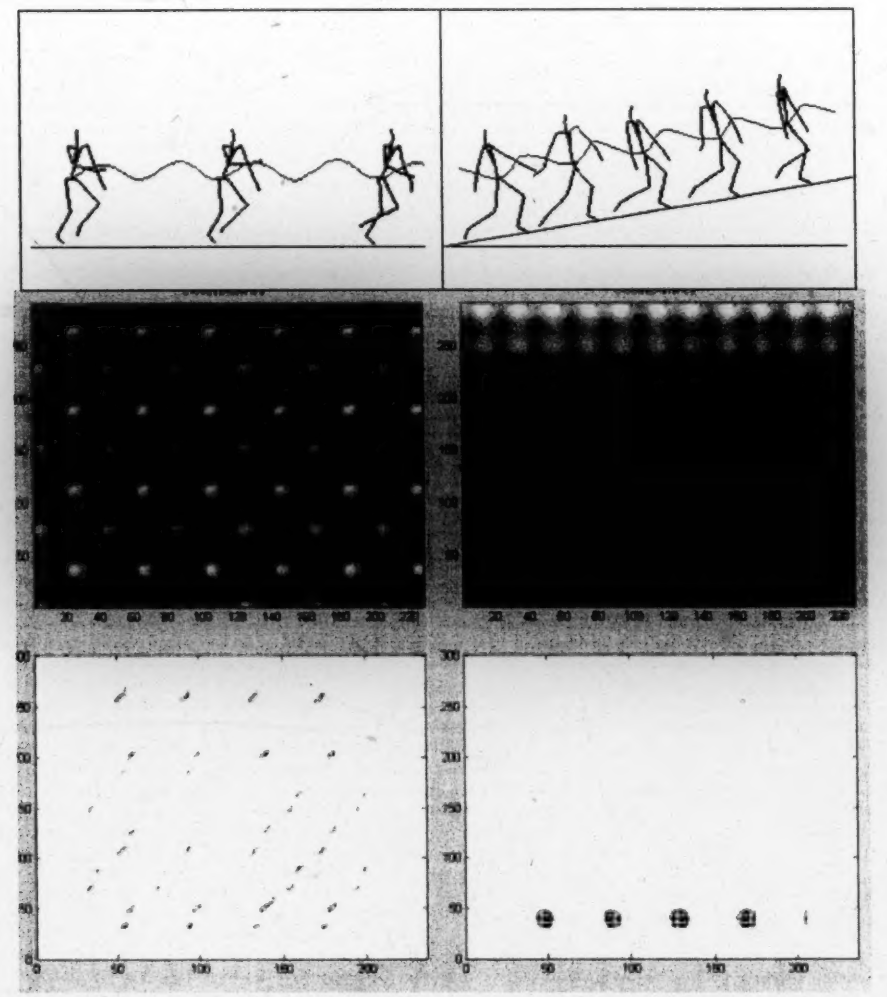


Figure 2.18: Top row: Two motion clips on the different level. Middle row and bottom row: Similarity analysis based on point clouds. Left: Two clips on the same level. Right: Two clips on the different level. Middle: The distance matrix. Bottom: The resulting similar frames.

### 2.6.3 Joint Weight

In the motion similarity metrics discussed above, different joint weights are used to control the importance of different joints. In real life, different joints surely have different visual importance to the perception or understanding of motion. For example, the hip joint, knee joint, shoulder joint, elbow, pelvis and spine are more important than the others. In [13], the important joint weights are set to one, while the unimportant ones are set to zero. J. Wang et al. [21] propose a set of optimal joint weights (shown in Table 2.1) for the similarity metrics proposed in [13]. They also compare their optimal joint weight set with the one in [13] by running a user study and found that the results using the optimal joint weight set are more robust and superior than the results using the original joint weight set in [13]. However, they indicate that their experimental results may be affected by the motion database and the different transition techniques applied. As all motion similarity analysis techniques have to specify a joint weight set used to control the importance of different joints, in the novel approach presented in the next chapter, we adopt the joint weight set as specified in [13].

| Joint name | Optimal joint weight |
|---|---|
| Right and Left Hip | 1.0000 |
| Right and Left Knee | 0.0901 |
| Right and Left Shoulder | 0.7884 |
| Right and Left Elbow | 0.0247 |

Table 2.1: The optimal joint weight set in [21]. Only the joints with non-zero weights are shown.

# Chapter 3

# A Novel Approach for Motion Similarity Analysis

Motion similarity analysis is an integral step in the reassembling-based synthesis approach, and has direct effects on the qualities of the resultant motion sequence. Moreover, in the previous similarity analysis methods, there are problems that are not addressed properly. In this chapter, we present a novel motion similarity analysis to help reusing motion capture data. But first we discuss how the two new features are selected in our approach.

## 3.1 Feature Selection

In this research, we are concerned with finding visually similar frames in two motion streams automatically. This means that the skeletal poses, joint velocities and accelerations of these two frames and of their neighboring frames should be similar enough. Thus the smooth and natural transitions between these two frames are able to be generated. In order to efficiently and accurately identify corresponding similar frames in two motion clips, suitable features are required to satisfy the following criteria.

- Coordinate invariant. This is important because the motion should not change after 3D rigid transformations. It is also called translation and rotation invariants. Here, we only consider the motion data with the same skeletal size.

- Efficient computation. Since similarity analysis is performed on a frame by frame basis, the computation time is at least $\mathbf{O}(n^2)$, where $n$ is the frame number in the motion database. For this reason, the feature computation cost is expected to be lower. Thus, the similarity computation can scale well with large motion databases.
- The similarity of skeletal pose and kinematic information should be incorporated.
- The method should not require determining the attribute weight parameters.

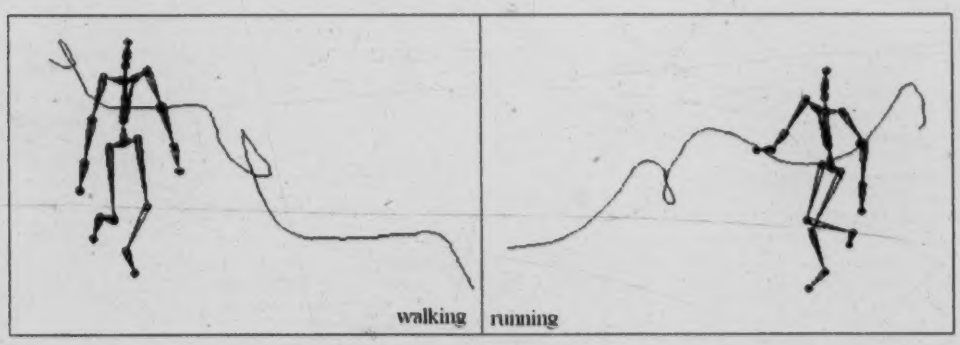



Figure 3.1: Two space curves formed by the movement of the LeftLowArm joint in two walking sequences.

Our goal is to find new motion features satisfying the above criteria. The movement of a joint forms a space curve, and a motion stream can be represented as a set of space curves formed by the trajectories of all the joints. This way, two motion sequences forms two sets of space curves. In Figure 3.1, two space curves are formed by the movements of the joint leftLowArm in two motion clips. From the viewpoint of space curve, the corresponding similar frames can be seen as two corresponding point sets in the two sets of space curves. From the theory of space curve, it is well known that curvature and torsion are the most important intrinsic properties of space curves. In the following section, we explore if these two properties can be used as the new features to detect frame similarities. First, the definitions of curvature and torsion are given by introducing Frenet formulae.

Typically, the parametric form of a space curve is represented as follows in Equation 3.1.

$$C(u) = [x(u), y(u), z(u)] \tag{3.1}$$

where $x(u), y(u)$ and $z(u)$ are the three components of curve $C(u)$, $u$ is a function of arc length $s$ of the curve.

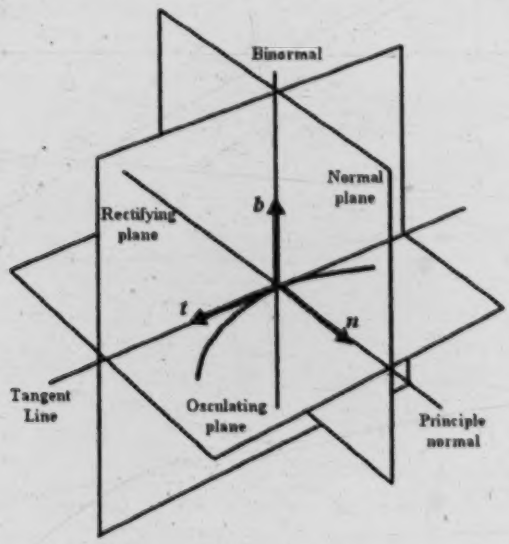



Figure 3.2: The Frenet frame of space curve

Every point $P$ of a space curve is associated with an orthogonal triple of unit vectors: the tangent vector $\mathbf{t}$, the principal normal vector $\mathbf{n}$ and the binormal vector $\mathbf{b}$ (Figure 3.2). They are defined as follows. The osculating plane at $P$ is defined to be the plane with the highest order of contact with the curve at $P$. The principal normal vector $\mathbf{n}$ is the vector perpendicular to the curve at $P$ and lies on the osculating plane. The binormal vector $\mathbf{b}$ is formed by the cross product of $\mathbf{t}$ and $\mathbf{n}$. Thus the frame formed by these three unit vectors is called the Frenet Frame at point $P$. The plane where $\mathbf{n}$ and $\mathbf{b}$ lie on is called the normal plane, while the plane containing $\mathbf{b}$ and $\mathbf{t}$ is called the rectifying plane. The derivatives of $\mathbf{t}$, $\mathbf{n}$ and $\mathbf{b}$ with respect to the arc length parameter $s$ give the following formulae.

$$\frac{\mathbf{dt}}{ds} = \kappa \mathbf{n}$$

$$\frac{\mathbf{dn}}{ds} = -\kappa\mathbf{t} + \tau\mathbf{b}$$

$$\frac{\mathbf{db}}{ds} = -\tau\mathbf{n}$$

These formulae are called the Frenet formulae of space curve. The coefficients $\kappa$ and $\tau$ are called the curvature and torsion of the curve, respectively. By definition, curvature represents the rate of change of the tangent vector, while torsion represents the rate of change of the osculating plane (or binormal vector $\mathbf{b}$) with respect to the arc length. Their geometric interpretation suggests that curvature and torsion reflect the tendency of change of space curve at a point. For that reason, these two intrinsic properties of space curve can be used to represent the movement information at that point (frame). However, the following examples suggest that more features are required for motion similarity analysis.

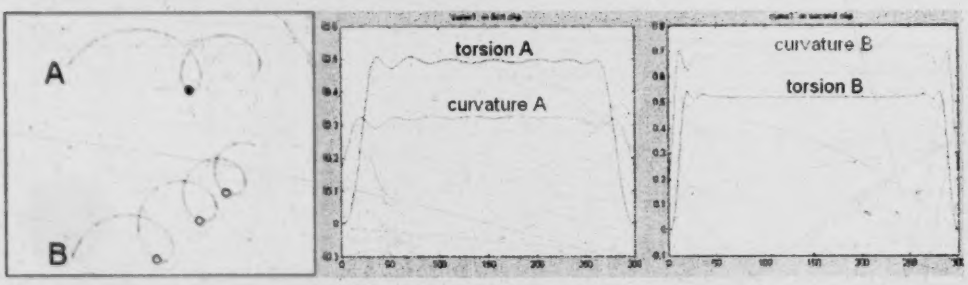



Figure 3.3: Left: Two spiral curves. Middle: Curvature and torsion for Curve A. Right: Curvature and torsion for Curve B.

Given two spirals defined as follows.

$$A : [\cos(t/\sqrt{2}), \sin(t/\sqrt{2}), t]$$

$$B : [\cos(2t/\sqrt{2}), \sin(2t/\sqrt{2}), t]$$

On each spiral, the curvature and torsion at each point are constant except at the boundaries (shown in Figure 3.3). Note: As B-splines are used to approximate the spirals, the boundary values of the curvatures and torsions are not reliable. We can say that any two frames of the two spirals match in terms of curvatures or torsions. In fact, the blue dot on Curve $A$ is expected to match with the yellow dots on Curve $B$. It is due to the fact that these two features only capture motion information. Therefore, another feature is required to capture the joint position difference. In

this research, the joint relative position with respect to its parent is selected to represent the joint position information. Moreover, the torsion computation involves third order derivatives, which need high-order spline approximation to compute it. Thus, we only use the curvature to capture the movement information that can be computed using cubic B-splines.

## 3.2 Similarity Analysis

Two novel features are used in our similarity analysis method. The first is the joint relative position in its parent's coordinate system. The second is the curvature of the global curve formed by the movement of each joint. Both of them are coordinate independent and can be easily computed from the motion capture data. In the approach developed in this thesis, there are two stages as shown in Figure 3.4, which are described in detail in the following sections.

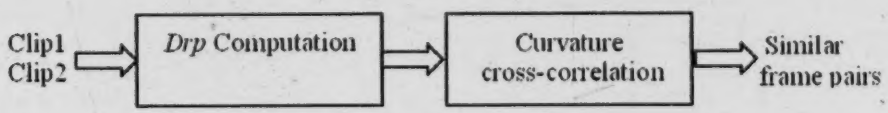



Figure 3.4: Motion similarity analysis process.

### 3.2.1 $D_{rp}$ Computation

The joint (except for the root joint) relative position and its difference in two motion frames are computed according to Equation 3.2 and 3.3.

$$\mathbf{rp} = R_z R_x R_y \mathbf{v} \tag{3.2}$$

$$D_{rp_{i,j}} = \sum_{k=1}^{m} w_k \|\mathbf{rp}_{i,k} - \mathbf{rp}_{j,k}\|^2 \tag{3.3}$$

$\mathbf{v}$ is the vector that represents the offset of a joint, $R_z, R_x$ and $R_y$ are the rotation matrices of its parent; $i$ and $j$ are the frame indices; $k$ is the joint index; $m$ is the number of joints; $w_k$ is the joint weight to control the importance of the $kth$ joint. Note that in this feature, the root joint (pelvis) is excluded. Since the root joint's parent is the global world, its relative position is the same as the root's global position. If the root joint were included in the difference sum, then a coordinate dependent factor would be introduced. The joint relative position reduces the effects of its parent's movements on the joint. The difference of this feature represents the position difference after the origins of their parents' coordinates have been aligned and hence, can capture the pose similarity more accurately. After this step, only the candidate frames that are similar enough will go to the second stage. Figure 3.5 shows the $D_{rp}$ matrix and candidate similar frame pairs between walking and running.

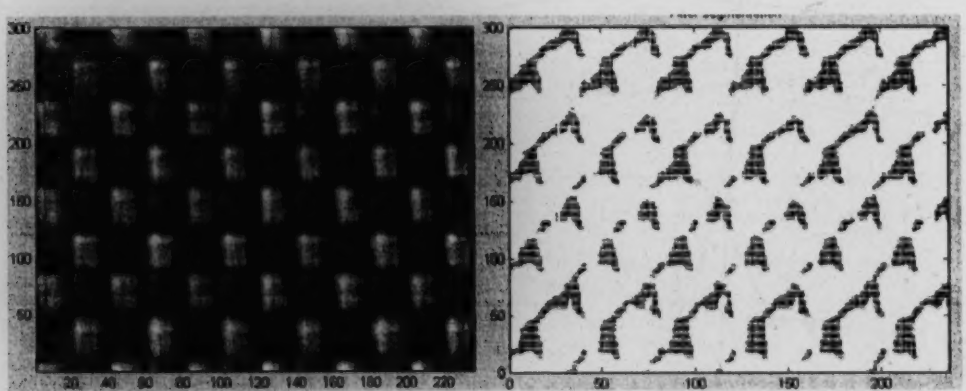


Figure 3.5: $D_{rp}$ computation between walking and running. Left: $D_{rp}$ matrix. Right: candidate similar frame pairs.

### 3.2.2 Curvature Cross Correlation

In the second stage, the global motion curve of each joint is generated and its curvatures are calculated. Then cross-correlation is used to find the corresponding similar frames based on the curvature information.

The space curve of a joint is calculated as given in Equation 3.4. The offset

vector of joint $\mathbf{v}$ is multiplied by the transformation matrix of its parent, then the transformation matrix of its grandparent and so on, until the transformation matrix of the root is reached. Each transformation matrix involves translations and rotations along each axis. This operation is performed in the same manner as that in determining the joint positions in the global coordinate frame.

$$\mathbf{r}(t) = M_{root} \dots M_{grandparent} M_{parent} \mathbf{v} \tag{3.4}$$

$$M = T R_z R_x R_y$$

When a space curve is represented in parametric form, its curvature is calculated according to Equation 3.5.

$$\kappa = \frac{\| \dot{\mathbf{r}} \times \ddot{\mathbf{r}} \|}{\| \dot{\mathbf{r}} \|^3} \tag{3.5}$$

In coordinate form

$$\kappa = \frac{\sqrt{\|A\|^2 + \|B\|^2 + \|C\|^2}}{((\dot{x})^2 + (\dot{y})^2 + (\dot{z})^2)^{\frac{3}{2}}}$$

where

$$A = \begin{pmatrix} \dot{y} & \dot{z} \\ \ddot{y} & \ddot{z} \end{pmatrix} \quad B = \begin{pmatrix} \dot{z} & \dot{x} \\ \ddot{z} & \ddot{x} \end{pmatrix} \quad C = \begin{pmatrix} \dot{x} & \dot{y} \\ \ddot{x} & \ddot{y} \end{pmatrix}$$

$\dot{\mathbf{r}}$ and $\ddot{\mathbf{r}}$ are, respectively, the first and second order derivatives of the space curve, respectively. $\dot{x}, \dot{y}, \dot{z}, \ddot{x}, \ddot{y}$ and $\ddot{z}$ are similarly defined. Since the motion capture data is represented in discrete form, the space curve formed by the joint's trajectory is represented as a set of discrete 3D points $P_0, P_1 \dots P_n$. In order to get the curvature of the motion curve of a joint, cubic B-splines are used to approximate the intervals between 2 sequential points.

The curve between two control points $P_i$ and $P_{i+1}$ is only affected by its 4 nearest points: $P_{i-1}, P_i, P_{i+1}$ and $P_{i+2}$ (shown in Figure 3.6). Note the curve does not pass through the control points. Given $n+1$ points, there will be $n-2$ cubic spline segments. The spline curve between $P_i$ and $P_{i+1}$ is defined in [9] by Equation 3.6 or 3.7, where $t_i \leq t < t_{i+1}$.

$$\mathbf{r}(t) = \frac{1}{6} \begin{pmatrix} (t-t_i)^3 & (t-t_i)^2 & t-t_i & 1 \end{pmatrix} \begin{pmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 0 & 3 & 0 \\ 1 & 4 & 1 & 0 \end{pmatrix} \begin{pmatrix} P_{i-1} \\ P_i \\ P_{i+1} \\ P_{i+2} \end{pmatrix} \tag{3.6}$$

Figure 3.6: B-spline approximation between two control points $P_i$ and $P_{i+1}$.

$$\mathbf{r}(t) = A(t-t_i)^3 + B(t-t_i)^2 + C(t-t_i) + D \tag{3.7}$$

where

$A = \frac{1}{6}(-P_{i-1} + 3P_i - 3P_{i+1} + P_{i+2})$

$B = \frac{1}{6}(3P_{i-1} - 6P_i + 3P_{i+1})$

$C = \frac{1}{6}(-3P_{i-1} + 3P_{i+1})$

$D = \frac{1}{6}(P_{i-1} + 4P_i + P_{i+1})$

Since cubic B-spline has $C^1$ and $C^2$ continuity, the first and second order derivatives at any point of the space curve can be evaluated using Equation 3.8 and 3.9 respectively, where $t = t_i$.

$$\dot{\mathbf{r}}(t_i) = \frac{1}{6}(-3P_{i-1} + 3P_{i+1}) \tag{3.8}$$

$$\ddot{\mathbf{r}}(t_i) = \frac{1}{3}(3P_{i-1} - 6P_i + 3P_{i+1}) \tag{3.9}$$

Before the B-spline approximation, the motion curves of joints are smoothed by a Gaussian filter to make the curvature values more stable. First, a suitable kernel is computed according to the Gaussian distribution function defined in Equation 3.10. $\sigma$ is the standard deviation of the distribution that determines the degree of smoothness. In our experiment, the kernel size is 9 and $\sigma$ is 2. Then the smoothing is achieved by a 1-D convolution operator. In Figure 3.7, the left image shows the convolution kernel used in our implementation, and the right image demonstrates the difference between the results of curvature with and without Gaussian smoothing.

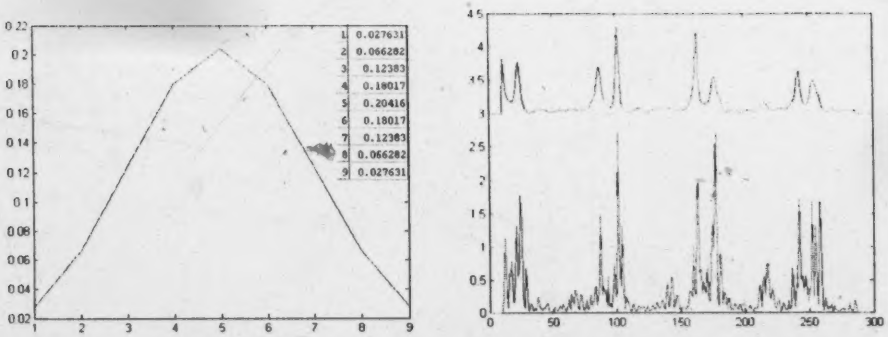



Figure 3.7: Left: The kernel used in Gaussian smoothing. ($\sigma = 2$ , $size = 9$) Right: The curvature of the motion curve for the joint leftLowArm in a walking clip, where the top signal is the curvature (shifted up by 3) calculated with a Gaussian filter, and the bottom signal is the curvature calculated without a Gaussian filter.

$$G(x) = \frac{1}{\sqrt{2\pi}\sigma} \exp(-\frac{x^2}{2\sigma^2}) \tag{3.10}$$

Curvature is an unsigned value that measures how rapidly the curve pulls away from its tangent. Therefore, it is used to represent the kinematic information of joint movements. The curvatures of each joint at frame $t$ form a feature vector for one motion stream. The dimension of the vector is determined by the number of joints.

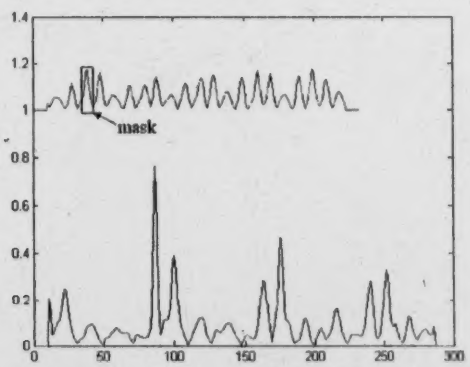



Figure 3.8: Cross correlation operated on the feature vectors of two motion clips. One dimensional feature formed by the trajectories of the left shoulder between a running and a walking sequence.

To identify similar movement pattern, cross-correlation is used, which is a standard method of estimating how two signals are correlated. It is a widely used technique in computer vision to identify similar 2D patterns. Here, the feature vector defined above in the neighborhood of a frame forms a pattern mask. The mask slides over the second motion clip. If a similar pattern can be found in the second clip, then the corresponding frame is similar to the frame in the middle of the mask in terms of curvature (see Figure 3.8). The cross correlation between two frames $i$ and $j$ is defined as in Equation 3.11.

$$corr_{i,j} = \sum_{k=1}^{m} w_k \sum_{l=-N/2}^{l<N/2} (\kappa_{i+l,k} - mean_{i,k})(\kappa_{j+l,k} - mean_{j,k}) \tag{3.11}$$

where $N$ is the size of the neighborhood, $w_k$ is the joint weight to control the importance of the $kth$ joint. Frame $i$ and its eight neighbors form a mask. We have found empirically that 9 frames is suitable for the size of the mask, since most transitions are in a short interval, ranging from 10 to 30 frames. $\kappa_{i+l,k}$ is the feature's $k^{th}$ dimension value of the $l^{th}$ neighbor of frame $i$; $mean_{i,k}$ is the feature's $k^{th}$ dimension mean value of frame $i$. $\kappa_{j+l,k}$ and $mean_{j,k}$ are similarly defined for frame $j$. According to the cross correlation result, the corresponding frame pairs with larger correlation values are considered as similar frame pairs. The left image of Figure 3.9 shows the cross correlation result between a walking and a running sequence, where the whiter areas have larger cross correlation values. Note that only the cross correlation results for the candidate frame pairs are computed, while the others are assigned the minimum of the computed cross correlation values to have the gray scale drawing. The right image in Figure 3.9 shows similar frame pairs as the output of our system.

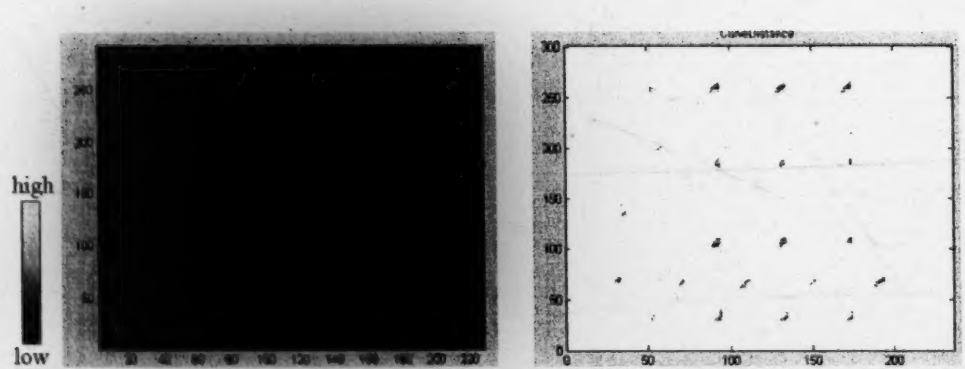



Figure 3.9: Left: Cross correlation results. Right: Similar frame pairs.

# Chapter 4

# Experimental Results

In this chapter, we present the experimental results on motion similarity analysis using the novel approach described in the previous chapter. The other 3 previous approaches for motion similarity analysis are also implemented. Then experiments designed to compare the performance of different methods are carried out. We focus on three major performance aspects: efficiency, accuracy and visual acceptability. In this research, three different ways are used to represent joint orientations, ie. Euler Angles, rotation matrices and quaternions. Appendix B contains their definitions and the conversion algorithms between them. All approaches are implemented on an average PC (1.16 GHz P4 with 2 GB of memory running Windows XP Professional) using C++, OpenGL and Maya 5.0. The motion dataset includes a wide variety of human motions, such as walking, running, dancing with different styles, and on different levels.

## 4.1 Run Time Comparison

Table 4.1 shows the run time comparison between different approaches, where the similarity computation time between walking (300 frames) and running (236 frames) is shown. Given a motion dataset involving $n$ clips, the total time needed for motion similarity analysis is Run Time (in Table 4.1) $\times \frac{n(n+1)}{2}$, when the number of frames for each motion clip is in the same range, that is, from 230 to 300 frames. The run time of the different motion similarity analysis methods ordered from fast to slow is: curvature and $D_{rp}$, joint positions and velocities, joint angles and velocities, and

point clouds. The results demonstrate that our approach is very efficient and can significantly improve the run-time performance for motion similarity analysis.

| Approaches | Run Time (seconds) |
|---|---|
| curvature and $D_{rp}$ | 1.842 |
| joint positions and velocities | 7.585 |
| joint angles and velocities | 9.844 |
| point clouds | 65.422 |

Table 4.1: Run time to compute the similarity between two motion clips for different motion similarity analysis methods.

## 4.2 Accuracy Study

### 4.2.1 Accuracy Criteria

From the proceeding discussion, it is already known that similar frame pairs should match in terms of skeletal configurations, joint velocities and joint accelerations. Thus in our experiments, three criteria are designed to evaluate the qualities of the results between different similarity analysis methods. They are, respectively, the velocity difference, the acceleration difference and the position difference ($D_v$ $D_a$ and $D_p$ for short), and defined in the following equations.

$$D_v = \sum_{i,j} \sum_{k=1}^{m} w_k \|\mathbf{v}_{i,k} - \mathbf{v}_{j,k}\|^2 \tag{4.1}$$

$$D_a = \sum_{i,j} \sum_{k=1}^{m} w_k \|\mathbf{a}_{i,k} - \mathbf{a}_{j,k}\|^2 \tag{4.2}$$

$$D_p = \sum_{i,j} \sum_{f=1}^{F} w_f \sum_{k=1}^{m} w_k \|\mathbf{p}^i_{k,f} - M_{i,0} M^{-1}_{j,0} \mathbf{p}^j_{k,f}\|^2 \tag{4.3}$$

where $\mathbf{v}$, $\mathbf{a}$ and $\mathbf{p}$ represent, respectively, the velocity, acceleration and position. $i$ and $j$ are the indices of the similar frame pair; $k$ is the joint index; $m$ is the number of joints; $w_k$ is the joint weight. $f$ is the frame index within the neighborhood of the compared frame; $F$ is the number of frames in the neighborhood; $w_f$ is the frame weight. $M_{i,0}$ is the transformation matrix of the root joint at frame $i$; $M^{-1}_{j,0}$ is the inverse of the transformation matrix of the root joint at frame $j$. When the

transition happens from frame $i$ to frame $j$, the transformation matrix ($M_{i,0}M_{j,0}^{-1}$) is used to align the two clips. Therefore, $D_p$ is able to roughly measure the similarity between the two compared frames and their neighborhoods. Actually, the physical interpretation of $D_p$ is similar to that of the minimal position difference in the point clouds approach. But they are different in the following two aspects. In our approach, a 3D transformation matrix is used to align the two point sets, while a 2D transformation matrix is used in the point clouds approach. Additionally, the 3D transformation matrix is directly computed from the transformation matrix of the root joints at the two compared frames, no other joints are involved. In the point clouds approach, a more complex computation scheme involving all the joints is carried out to compute the 2D transformation matrix, and makes the computation cost more expensive. For example, on the same PC, it takes about 65 seconds to compute the point clouds distance matrix between two clips ($300 frames \times 236 frames$), while it takes about 40 seconds to compute $D_p$ for the same two clips. Moreover, since the position difference $D_p$ can individually capture the similarity between two frames and their neighborhoods, it is more important than the other two measures, and is used as a major evaluation criterion in our evaluation scheme.

To compute the above three measures, the same number of similar frame pairs are generated between any two clips in the motion datasets for each approach. The quick sort algorithm in [8] is used to order the computed similarity measure between any two motion clips. The frame pairs with a higher similarity are kept as the results. For example, top 1 percent ($0.1 \sim 1.5$ percent in our experiments) of all the frame pairs are output as the resulting similar frame pairs. Note that the curvature and $D_{rp}$ approach sorts the candidate frame pairs according to the cross-correlation values in the second step.

### 4.2.2 Optimal Attribute Weights Computation

The measure $D_p$ can provide guidelines to find an optimal attribute weight set needed in the approaches based on the weighted sum of multiple joint attributes. At this point, the computed optimal attribute weights are used to control the relative importance of the different joint attributes, such as joint positions, joint angles, root velocities and accelerations. They are different from the joint weights discussed in

Section 2.6.3, which control the importance of different joints. In our experiments, we adopt the joint weight set as that specified in [13]. That is, The weights of the important joints are set to one, while the unimportant ones are set to zero. All the other approaches use the same joint weight set.

The process to compute the optimal attribute weights needed in the approaches based on the weighted sum of multiple joint attributes is described in detail as follows.

1. Design an initial attribute weight search space. For example, each weight has 10 possible values evenly distributed in the range $(0, 1]$.

2. For each possible attribute weight set, calculate $D_p$. Then the attribute weight set that results with the minimum $D_p$ corresponds to the optimal attribute weight set in the current search space.

3. Refine the attribute weight search space if necessary.

The exhausting search for an optimal attribute weight set is very time-consuming and the optimal attribute weight set depends on the motion patterns in the dataset. For example, in the approach based on joint angles and velocities, when the dataset includes only 2 clips (each has 150 frames), and the search space is $10 \times 10 \times 10$, it takes more than 4 hours to find the optimal attribute weights. When the search space increases to $20 \times 20 \times 20$, it needs about 34 hours. In the first dataset, which includes running and walking only, the optimal attribute weight set is $(0.1, 1.0, 0.1)$. In the second dataset, which includes dancing and running, the optimal attribute weight set becomes $(1.0, 1.0, 0.1)$. If no measures such as $D_p$ are defined to help determine the optimal attribute weights automatically, then the animators have to estimate the values of these weights by trial-and-error, which is obviously a very tedious and labor intensive exercise. Furthermore, the quality of the results depends on the experience and skills of the animator. So compared with these techniques, our approach has the major advantage that it does not require the animator to set any attribute weights.

### 4.2.3 Results for Accuracy Evaluation

In this section, the results for accuracy evaluation are presented when different motion similarity analysis techniques are applied to different datasets.

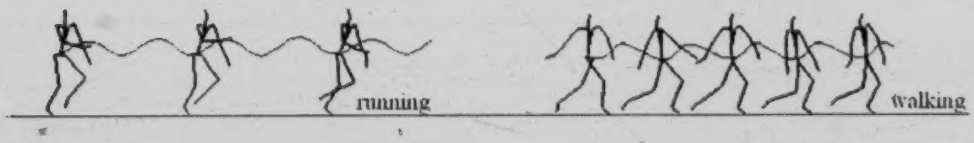



Figure 4.1: Two motion clips on the same floor plane.

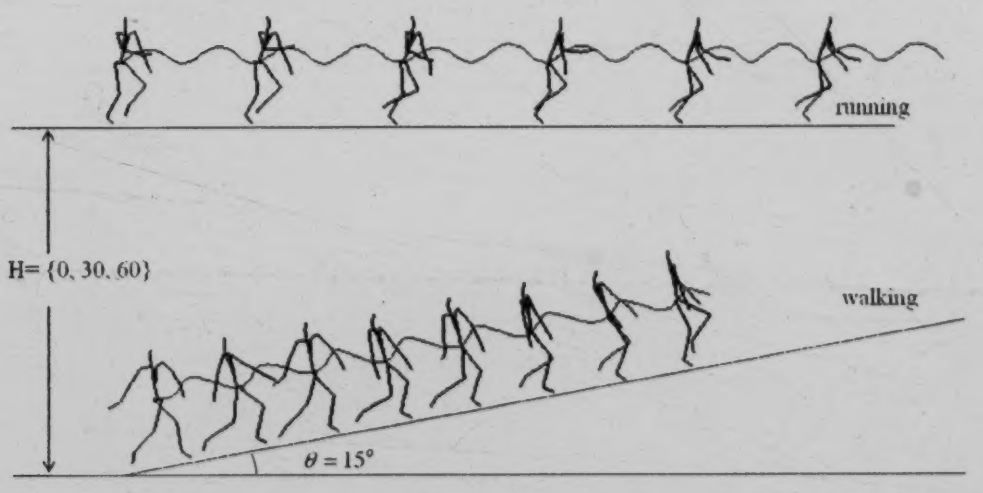



Figure 4.2: Two motion clips on different levels. Top: Running on the floor plane. Bottom: Walking along a ramp with a slope of 15 degrees. $H$ represents the vertical distance by which the walking clip is shifted down from the floor plane.

**Motions on Different Levels** The two clips are running and walking on the same floor plane shown in Figure 4.1. In the different level cases, the walking is modified by rotating the root joint around the z-axis by 15 degrees so that walking on the ground plane is changed to walking on a slope. Then two more walking clips (see Figure 4.2) on a ramp are generated by shifting the rotated walking clip down by 30 and 60 cm. Thus, the effects of the vertical distance between the root joints

of the two compared frames can be investigated.

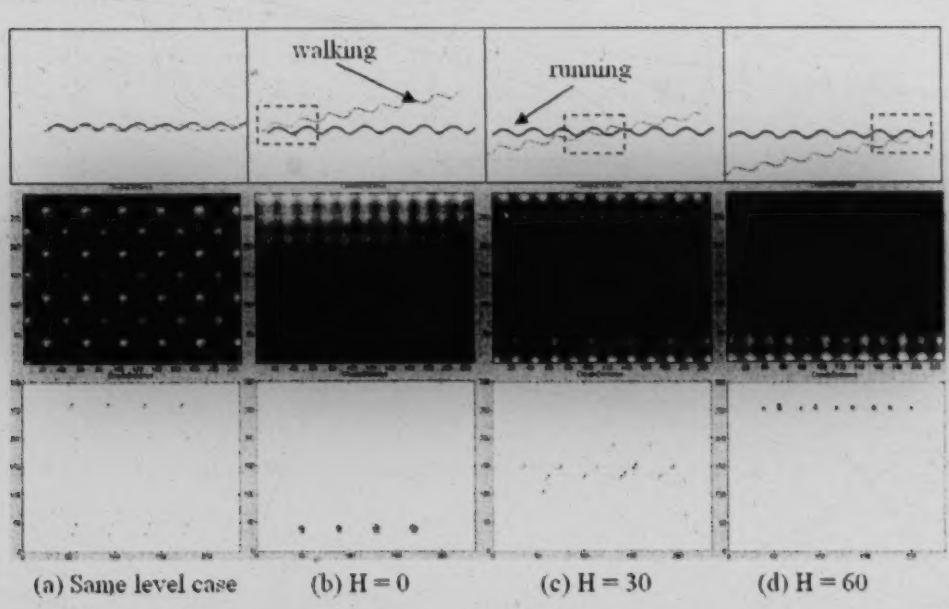
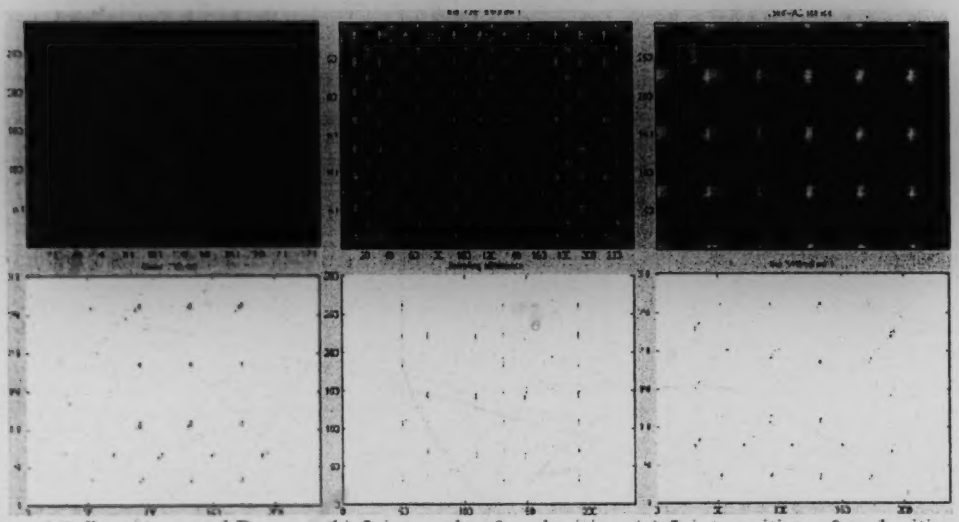



Figure 4.3: Similarity analysis based on point clouds. (a) two clips are on the same floor plane; (b)(c)(d) on 3 different level cases. Top: The root trails of walking and running. The dotted boxes mark the areas with a smaller vertical distance of the root joints. Middle: The distance matrix (darker areas imply a lower distance measure). Bottom: The resulting similar frame pairs denoted by blue dots. The rows represent the frames in the running clip, while the columns represent the frames in the walking clip.

The top images in Figure 4.3 show the trails of the root joints in walking and in running, where the areas with a smaller vertical distance of the root joints are marked by the dotted boxes. The middle and bottom images in Figure 4.3 show the similarity analysis results in the point clouds approach between the same level case and 3 different level cases. Since the point clouds approach uses a 2D rigid transformation matrix to compute the minimal distance between two point sets, the similarity results obtained by this method are quite different between the four compared cases. When the vertical distance between the root joints of the two clips increases, the difference measure increases, or the similarity measure decreases. While in the other three techniques, since all the motion features are coordinate independent, the same results are achieved between the same level case and the

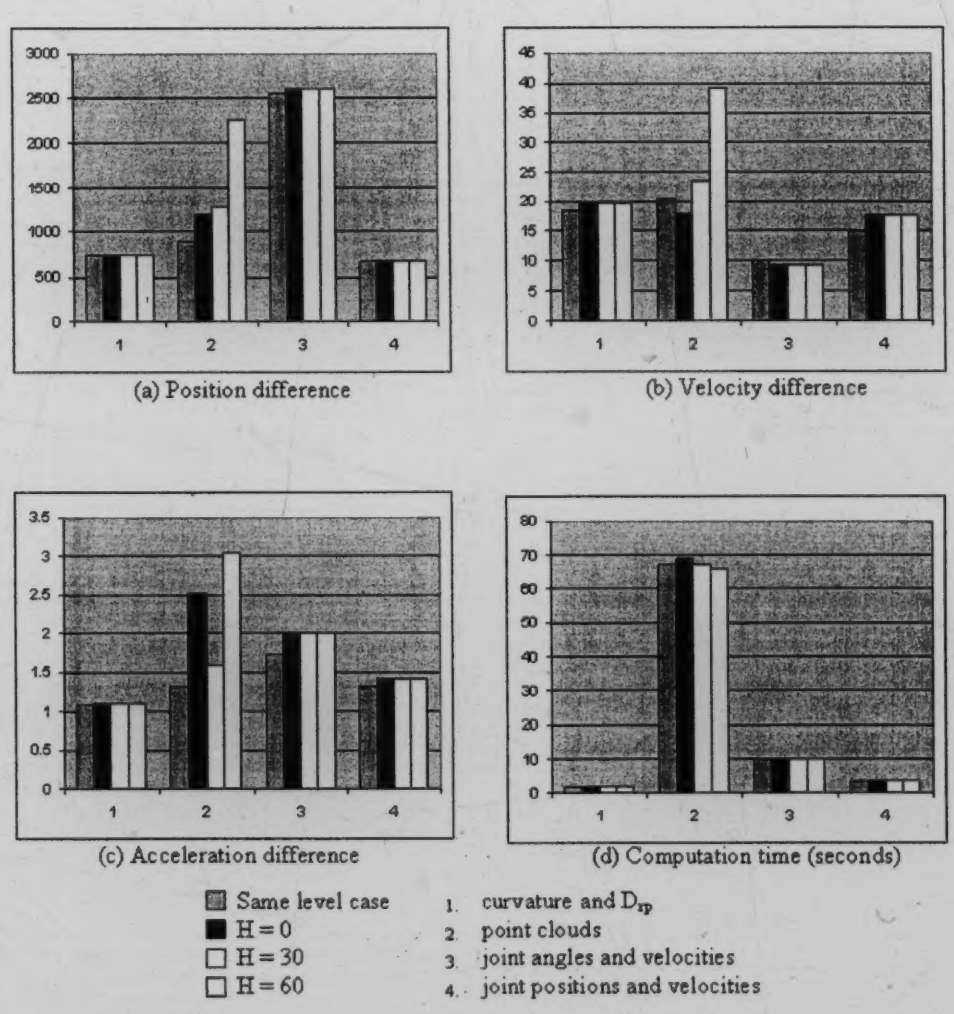


(a) Curvature and $D_{rp}$ (b) Joint angles & velocities (c) Joint positions & velocities

Figure 4.4: Similarity analysis based on: (a) Curvature and $D_{rp}$; (b) Joint angles and velocities; (c) Joint positions and velocities. Top left: Cross-correlation results. Top middle and right: Distance matrix. Bottom: The resulting similar frame pairs.

different level cases for each technique (shown in Figure 4.4). Figure 4.5 shows the performance comparison between the same level case and 3 different level cases. The results demonstrate that the accuracy performance of the point clouds approach decreases largely in the different level cases when compared with that in the same level cases; while the other approaches keep the same performance in both cases. Here, in the approaches based on the weighted sum of multiple joint attributes, all the weights are set to one. According to the criterion $D_p$, the performance of the approach based on curvature and $D_{rp}$, the approach based on joint positions and velocities, the point clouds approach in the same level case are much better than the performance of the approach based on joint angles and velocities. It is because the unit attribute weight set used in the joint angle approach is not suitable for the test motion clips.

**Optimal Attribute Weight Sets Computation** In the joint angle approach, three weights ($w_p : w_a : w_v$) are used to accommodate for the relative importance of the difference of root velocities, the difference of joint angles and the difference of joint velocities. To get a suitable attribute weight set needed in the joint angle




(a) Position difference

(b) Velocity difference

(c) Acceleration difference

(d) Computation time (seconds)

Figure 4.5: Performance comparison between the same level case and the 3 different level cases. In the different level case, the performance for the point clouds approach decreases largely when compared with its performance in the same level case; while the other approaches perform almost the same in both cases.

approach, the search process described in Section 4.2.2 is operated in two search spaces. Both are $10 \times 10 \times 10$, one evenly distributed, and the other non-uniformly distributed. Two optimal attribute weight sets are obtained for the test motions. Figure 4.6 shows the difference in the results generated by using the joint angle approach with different attribute weight sets. In Figure 4.7, the performance for the approach based on joint angles and velocities is presented with different attribute weight sets. The results demonstrate that the performance with the optimal attribute weight sets is largely improved when compared with the user specified weights. In the attached CD, Movie 1 also shows the difference in the results by using a user specified attribute weight set $(1.0, 1.0, 1.0)$ and the improved optimal attribute weight set $(0.05, 1.0, 0.02)$.

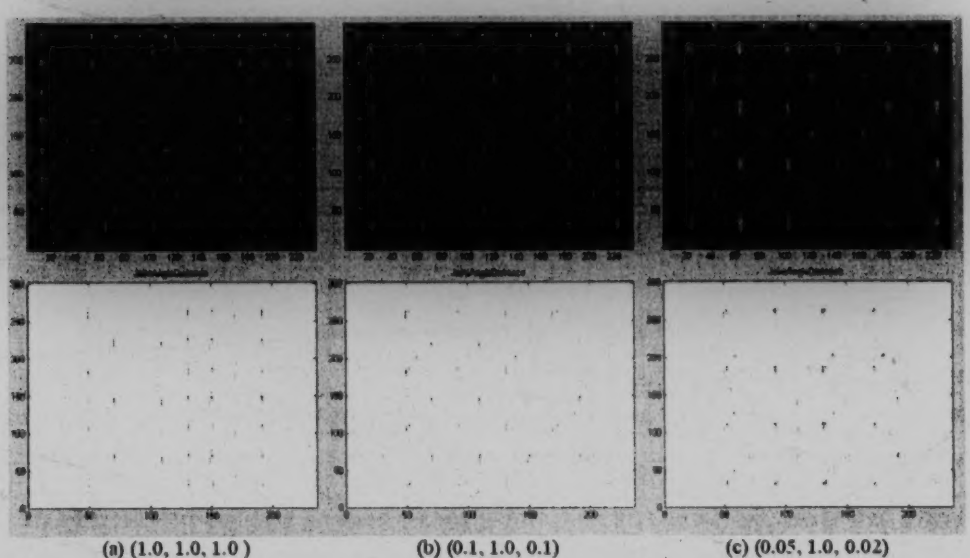
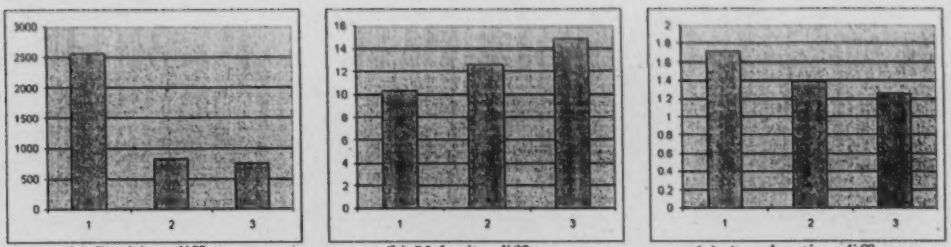



Figure 4.6: Similarity analysis based on joint angles and joint velocities with different attribute weight sets. Left: A user specified attribute weight set. Middle and right: Two optimal attribute weight sets. Top: The distance matrix. Bottom: The resulting similar frame pairs denoted by blue dots.

**Evaluation between Different Datasets** Three different datasets have been tested using different motion similarity analysis approaches. Set A includes a walking and running clip ($300 \times 236$ frames). Set B includes a rocknroll and a highwire walking ($260 \times 300$ frames). Set C has 12 motion clips with a variety of human



(a) Position difference. (b) Velocity difference. (c) Acceleration difference.

1. A user specified weight set { 1.0 : 1.0 : 1.0 }
2. Optimal attribute weight set One { 0.1 : 1.0 : 0.1 }
3. Optimal attribute weight set Two { 0.05: 1.0 : 0.02}

Figure 4.7: Performance comparison for the joint angle approach by using different attribute weight sets. The performance of the joint angle approach is largely improved by using the computed optimal attribute weight sets according to $D_p$.

behaviors, such as sneaking, drunk-walking and dancing. For the three datasets, all the motions happen on the same level ground and the joint angle approach is applied with an improved attribute weight set. Figure 4.8 shows the similarity analysis results for Set B. The performance of different approaches for the 3 datasets is presented in Figure 4.9. The result demonstrates that the performance of different approaches varies a little when different datasets are applied. However, according to the position difference criterion, there is not too much difference in their performance in terms of accuracy or quality. This can be verified by the similarities of their performance in terms of visual acceptability presented in the next section. Furthermore, the whole evaluation process shows that the two criteria, the velocity difference ($D_v$) and the acceleration difference ($D_a$), cannot be independently used as an evaluation criterion.

The results of our evaluation between different approaches are summarized in Table 4.2.

## 4.3 Visual Acceptability

To visually assess the performance of the similarity analysis results, a hierarchical graph is built based on the results of different motion similarity analysis methods.

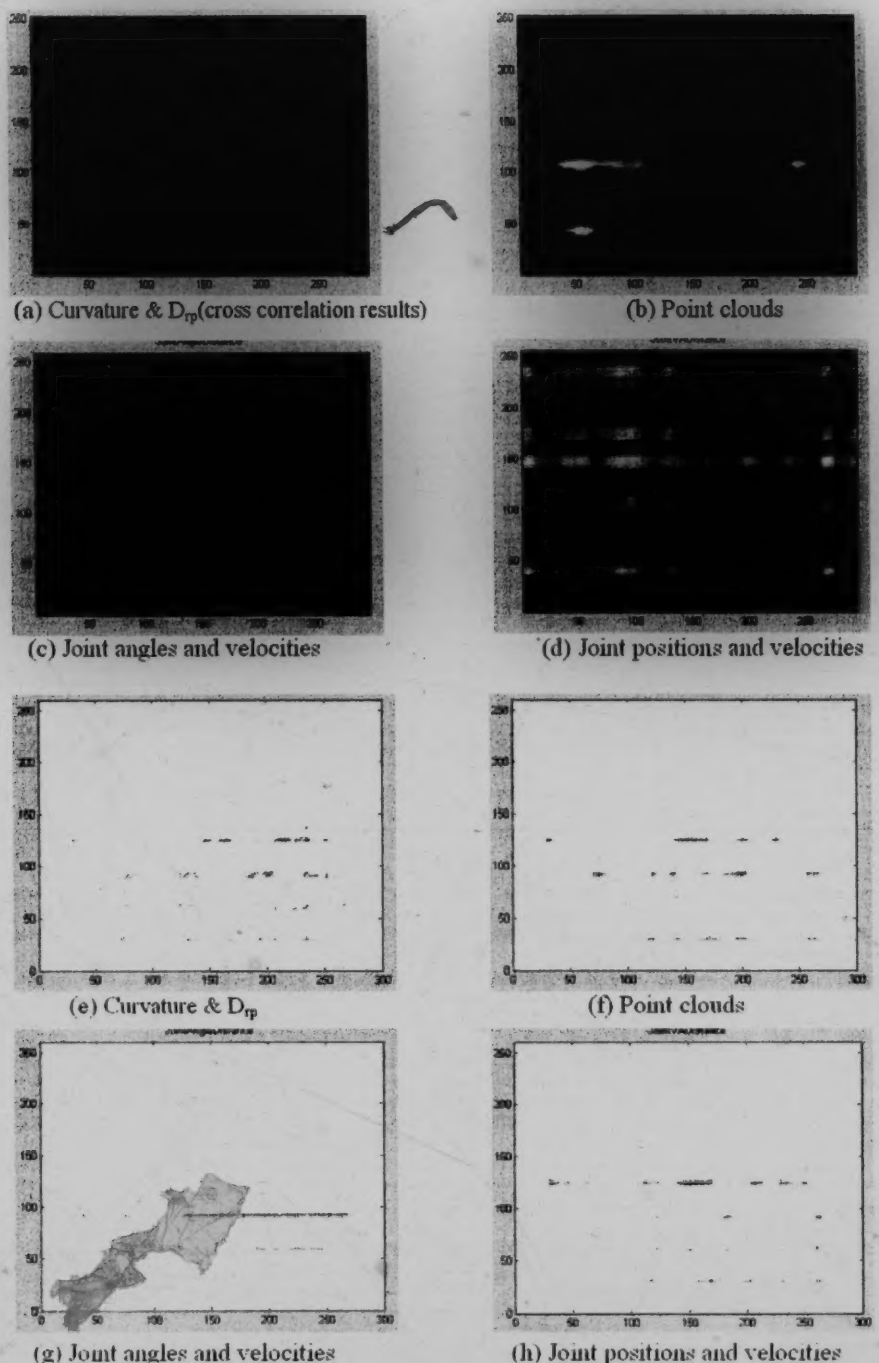
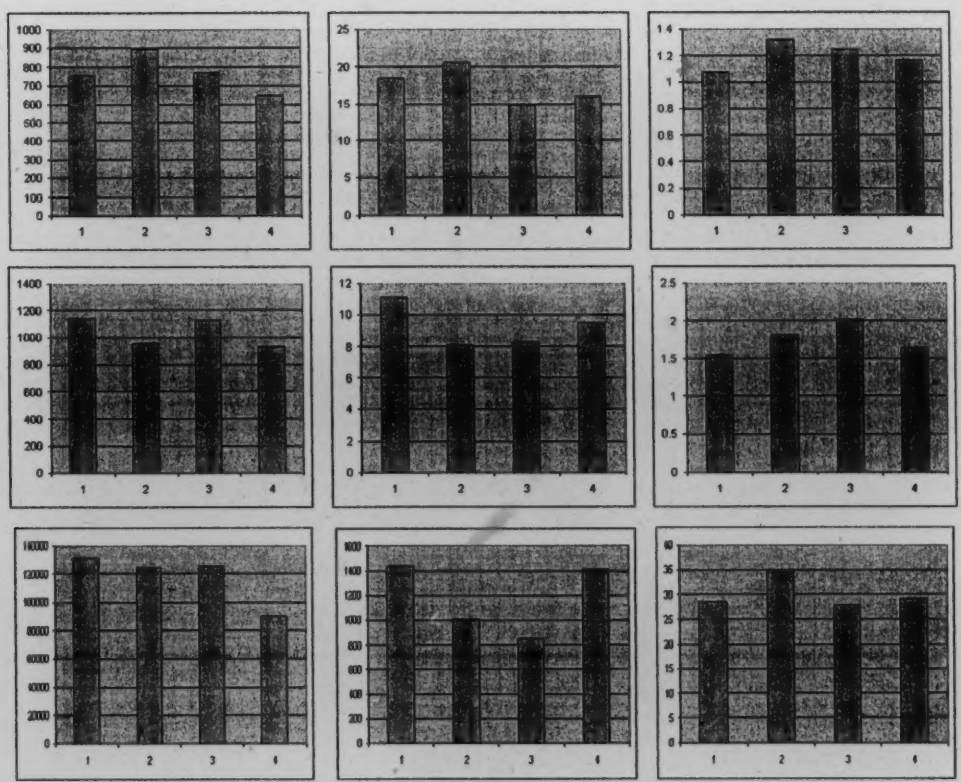

(a) Curvature & $D_{rp}$(cross correlation results)
(b) Point clouds
(c) Joint angles and velocities
(d) Joint positions and velocities
(e) Curvature & $D_{rp}$
(f) Point clouds
(g) Joint angles and velocities
(h) Joint positions and velocities

Figure 4.8: Motion similarity analysis results for Dataset B. (a) the cross correlation results of our approach; (b)(c)(d) the distance matrices; (e)(f)(g)(h) the resulting similar frame pairs.



(a) Position difference. (b) Velocity difference. (c) Acceleration difference.

1. Curvature & $D_{vp}$
2. Point clouds
3. Joint angles & velocities
4. Joint positions & velocities

Figure 4.9: Performance comparison between 3 different datasets. Top row: Dataset A, a walking and a running ($300 \times 236$ frames). Middle row: Dataset B, a rocknroll and a highwire walking ($260 \times 300$ frames). Bottom row: Dataset C, 12 different motion clips. (a) the position difference; (b) the velocity difference; (c) the acceleration difference.

Then according to the user's specifications, new motion sequences are created by stitching different motion clips together. Since motion transition points are detected based on motion similarity analysis, the quality of the resulting motion sequence directly depends on the similarity analysis results. In our experiments, linear interpolation is applied to distribute the discontinuities in the transition region. The experimental results demonstrate that the smooth and natural transitions can be achieved at the similar frame pairs generated by using the curvature and $D_{rp}$ approach. Figure 4.10 shows the transitions between walking and running. The left columns show the skeleton poses at selected similar frame pairs. The middle and right columns are the transition results without and with linear smoothing. In the attached CD, Movie 2 to Movie 4 demonstrate that our approach can generate visually appealing transitions. Movie 5 shows the transition results based on the other similarity analysis techniques.

| Evaluation | Curvature and relative positions | Point clouds | Joint angles and velocities | Joint positions and velocities |
|---|---|---|---|---|
| Run Time | Fast | Slow | Medium | Medium |
| Optimal Attribute Weight Determination | No | No | Yes | Yes |
| Coordinate Invariant | Yes | No | Yes | Yes |
| Velocity Difference | Acceptable | Good | Good | Acceptable |
| Acceleration Difference | Good | Acceptable | Good | Good |
| Position Difference | Acceptable | Acceptable | Acceptable | Good |

Table 4.2: Comparison between different approaches for motion similarity analysis.

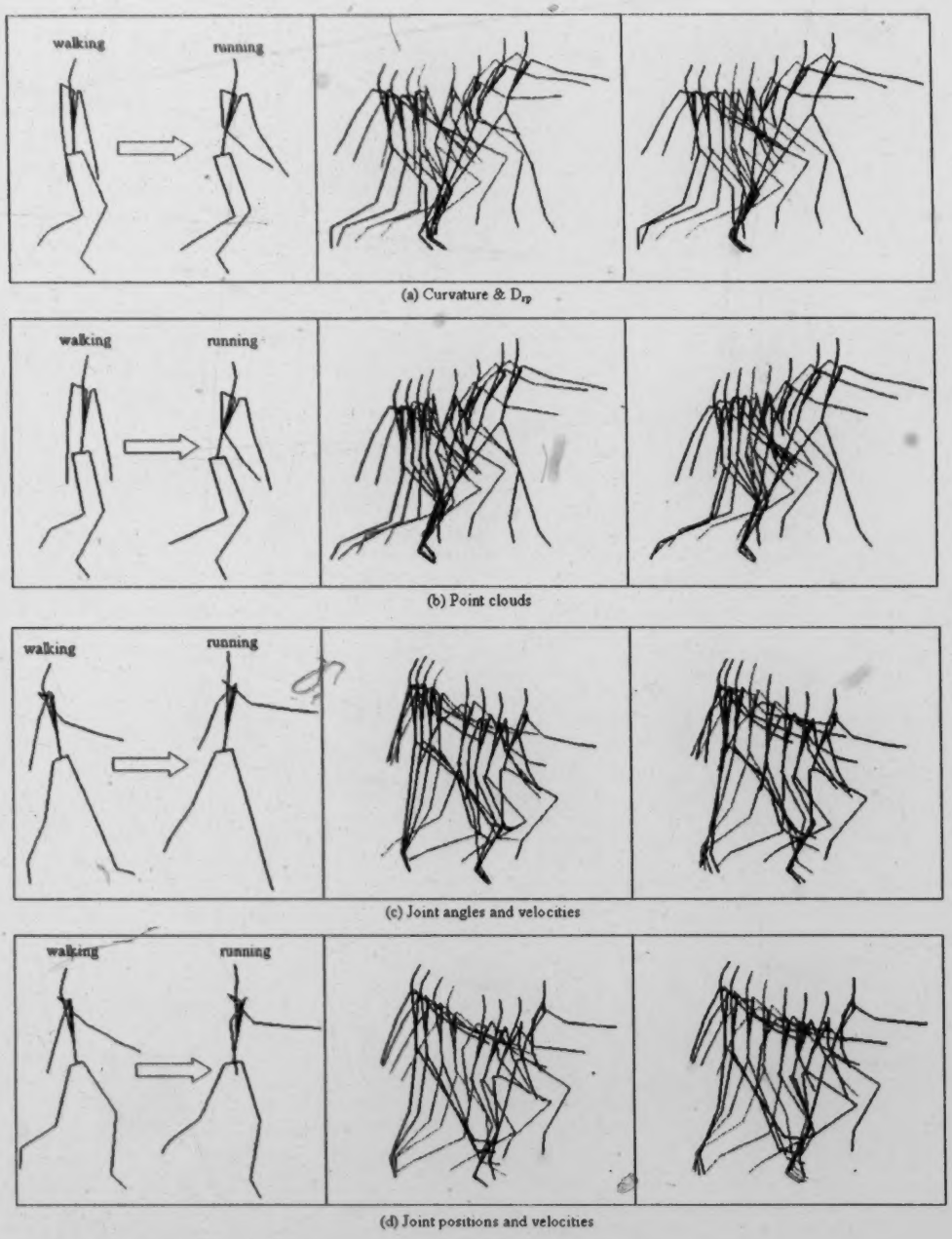



Figure 4.10: Transitions between walking (green) and running (red). Left: Skeleton poses at the selected similar frame pairs. Middle: Transitions without linear interpolation. Right: Transitions with linear interpolation. The screen shots are down-sampled by a factor of 3.

# Chapter 5

# Conclusions and Future Work

## 5.1 Contributions

In this thesis, two typical motion similarity approaches are investigated and the related problems are identified. The approach based on a weighted sum of multiple joint attributes requires attribute weight determination, while the point clouds approach is coordinated variant and not efficient. To tackle the problems existed in the previous approach, a novel method for motion similarity analysis is designed and developed. Two motion features: the curvature of space curve and the joint relative positions are presented to estimate the similarity between two motion frames. The experimental results show that our approach is very efficient and is completely coordinate invariant. And visually acceptable transition results can be generated. We also introduce a general criterion to evaluate the performance of different methods for motion similarity analysis. By using this criterion, better attribute weight parameters can be found to improve the results of the approach based on the weighted sum of different joint attributes.

For the approach based on joint positions, velocities and accelerations, as the similarity is represented as the weighted sum of three or more components, the appropriate attribute weight set is very difficult to determine. Moreover, for the dataset with different motion patterns, the attribute weight set is likely to change. In our approach, although two motion features are used, the two-step process instead of the weighted sum avoids the attribute weight selection. Meanwhile, in the case of the point clouds approach, the minimum distance between two point sets is computed by aligning them with a 2D rigid transformation matrix. As a result, this method

cannot correctly deal with motions on different plane levels. The universal criterion $D_p$ presented in this thesis is different from the minimum distance in the point clouds approach for the following two reasons. First, $D_p$ is the distance after two point sets are aligned by a 3D transformation matrix instead of a 2D transformation matrix. Second, the 3D transformation matrix is directly computed from the 6 DOF of the root joints of the two frames. Therefore, $D_p$ is more general and easily computed. And according to $D_p$, an optimal attribute weight set can be found for approaches that require attribute weight determination.

In our approach, two novel motion features are proposed to describe the similarity between two motion frames. The joint relative positions capture the skeletal pose information, while the curvature of space curve formed by the joint movements capture the kinematic information. Compared with the previous approach as for motion similarity analysis, the new curvature and $D_{rp}$ approach has three major advantages. First, it has a high efficiency. It is because the two features can be computed directly from motion capture data. Second, it is coordinate invariant. The two selected features are coordinate independent. Thus, the proposed approach can correctly handle a wide range of motions, particularly, the motions on different levels. Finally, no attribute weight determination is required in our approach. A two-step process does not require attribute weight selection.

There is one limitation of the curvature and $D_{rp}$ approach. The accuracy of the curvature computation may affect the quality of the similarity analysis results. But it can be improved by using the ENO schemes (Essentially Non-Oscillatory) that are introduced in [7].

In summary, three major original contributions have been made and described in this thesis:

1. The problems related with current approaches for motion similarity analysis are identified.

2. A novel approach using two motion features are proposed to estimate the similarity between two motion frames. It has the following advantages:

   - high efficiency,
   - coordinate invariant, and

- no attribute weight determination.

3. A novel criterion is presented to compare the performance of different approaches and to compute the optimal attribute weight for the approaches based on the weighted sum of different joint attributes.

## 5.2 Future Work

In this thesis, there are several areas left open for future work. One possibility is to improve the performance of the curvature and $D_{rp}$ approach by using a more accurate method for the curvature computation. In our implementation, the cross-correlation is performed in the spatial domain. In the future, we plan to investigate the effects when cross-correlation is performed in the Fourier domain. In theory, the FFT can further improve the efficiency.

As well, we would like to apply our work to much larger datasets with various motion patterns and to test the effects of using larger data sets on the performance of different approaches. Moreover, we will investigate if there are other motion features that are more suitable to represent and to identify motion similarities between two motion frames.

# Bibliography


[1] A. Ahmed, F. Mokhtarian, and A. Hilton. Parametric motion blending through wavelet analysis. short presentations. In *Proceedings of Eurographics*, pages 347–353, 2001.

[2] O. Arikan and D. A. Forsyth. Interactive motion generation from examples. In *Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, pages 483–490. ACM Press, 2002.

[3] M. Brand and A. Hertzmann. Style machines. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pages 183–192. ACM Press/Addison-Wesley Publishing Co., 2000.

[4] A. Bruderlin and L. Williams. Motion signal processing. In *Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, pages 97–104. ACM Press, 1995.

[5] M. Gleicher. Motion editing with spacetime constraints. In *Proceedings of the 1997 symposium on Interactive 3D graphics*, pages 139–ff. ACM Press, 1997.

[6] M. Gleicher. Retargetting motion to new characters. In *Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, pages 33–42. ACM Press, 1998.

[7] A. Harten, B. Engquist, S. Osher, and S. R. Chakravarthy. Uniformly high order accurate essentially non-oscillatory schemes, 111. *J. Comput. Phys.*, 71(2):231–303, 1987.

[8] http://linux.wku.edu/ lamonml/algor/sort/quick.html.

[9] http://www.euclideanspace.com/maths/geometry/curve/.

[10] http://www.theonering.net/scrapbook/view/6856.

[11] L. Kovar and M. Gleicher. Flexible automatic motion blending with registration curves. In *Proceedings of the 2003 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 214–224. Eurographics Association, 2003.

[12] L. Kovar, M. Gleicher, and F. Pighin. Motion graphs. In *Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, pages 473–482. ACM Press, 2002.

[13] J. Lee, J. Chai, P. S. A. Reitsma, J. K. Hodgins, and N. S. Pollard. Interactive control of avatars animated with human motion data. In *Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, pages 491–500. ACM Press, 2002.

[14] J. Lee and S. Y. Shin. A hierarchical approach to interactive motion editing for human-like figures. In *Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, pages 39–48. ACM Press/Addison-Wesley Publishing Co., 1999.

[15] Y. Li, T. Wang, and H.-Y. Shum. Motion texture: a two-level statistical model for character motion synthesis. In *Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, pages 465–472. ACM Press, 2002.

[16] S. I. Park, H. J. Shin, and S. Y. Shin. On-line locomotion generation based on motion blending. In *Proceedings of the 2002 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 105–111. ACM Press, 2002.

[17] K. Pullen and C. Bregler. Motion capture assisted animation: texturing and synthesis. In *Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, pages 501–508. ACM Press, 2002.

[18] C. Rose, M. F. Cohen, and B. Bodenheimer. Verbs and adverbs: Multidimensional motion interpolation. *IEEE Comput. Graph. Appl.*, 18(5):32–40, 1998.

[19] C. Rose, B. Guenter, B. Bodenheimer, and M. F. Cohen. Efficient generation of motion transitions using spacetime constraints. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pages 147–154. ACM Press, 1996.

[20] M. Unuma, K. Anjyo, and R. Takeuchi. Fourier principles for emotion-based human figure animation. In *Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, pages 91–96. ACM Press, 1995.

[21] J. Wang and B. Bodenheimer. An evaluation of a cost metric for selecting transitions between motion segments. In *Proceedings of the 2003 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 232–238. Eurographics Association, 2003.

[22] D. J. Wiley and J. K. Hahn. Interpolation synthesis of articulated figure motion. *IEEE Comput. Graph. Appl.*, 17(6):39–45, 1997.

[23] A. Witkin and Z. Popovic. Motion warping. In *Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, pages 105–108. ACM Press, 1995.

# Appendix A

# The Header Part of a BVH File

This appendix illustrates the header part of a standard BVH File.

```
Hips {
        LeftHip (LeftUpLeg) {
                LeftKnee (LeftLowLeg) {
                        LeftAnkle (LeftFoot) {
                                End Site {}
                                } } }
        RightHip (RightUpLeg) {
                RightKnee (RightLowLeg) {
                        RightAnkle (RightFoot) {
                                End Site {}
                                } } }
        Chest {
                LeftCollar {
                        LeftShoulder (LeftUpArm) {
                                LeftElbow (LeftLowArm) {
                                        LeftWrist (LeftHand) {
                                                End Site {}
                                                } } } }
                RightCollar {
                        RightShoulder (RightUpArm) {
                                RightElbow (RightLowArm) {
                                        RightWrist (RightHand) {
                                                End Site {}
                                                } } } }
                Neck {
                        Head {
                                End Site {}
                                } } } }
```

# Appendix B

# Joint Orientation Representations and Conversions

This appendix contains the definitions of the 3 joint orientation representations and their conversion algorithms used in this research. Three ways are used to represent joint rotations, namely, Euler Angles, rotation matrices and quaternions. In motion capture data, the orientation of a joint is recorded by using Euler Angles, in which three degrees of freedom along each axis in a fixed order are used to express the joint orientation. 3 series of rotations along each axis, $R_z, R_x, R_y$ are independently applied to the joint. As the 3D rotations are not commutative, the rotation order is important.

An alternative way to represent joint rotations is using a $3 \times 3$ matrix. In the right-handed coordinate system, the rotation around the z-axis by the angle $\theta_z$ is given by the following matrix.

$$R_z(\theta_z) = \begin{pmatrix} \cos(\theta_z) & -\sin(\theta_z) & 0 \\ \sin(\theta_z) & \cos(\theta_z) & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Similarly, the rotation around the x-axis and the y-axis are, respectively,

$$R_x(\theta_x) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos(\theta_x) & -\sin(\theta_x) \\ 0 & \sin(\theta_x) & \cos(\theta_x) \end{pmatrix}$$

$$R_y(\theta_y) = \begin{pmatrix} \cos(\theta_x) & 0 & \sin(\theta_z) \\ 0 & 1 & 0 \\ -\sin(\theta_z) & 0 & \cos(\theta_x) \end{pmatrix}$$

A general rotation matrix is obtained by a combination of the above three matrices defined as follows.

$$R = R_z R_x R_y$$

$$\begin{pmatrix} m_0 & m_1 & m_2 \\ m_3 & m_4 & m_5 \\ m_6 & m_7 & m_8 \end{pmatrix} = \begin{pmatrix} c_z c_y - s_z s_x s_y & -s_z c_x & c_z s_y + s_z s_x c_y \\ s_z c_y - c_z s_x s_y & c_z c_x & s_z s_y - c_z s_x c_y \\ -c_x s_y & s_x & c_x c_y \end{pmatrix}$$

where $c_z$ denotes $\cos(\theta_z)$, and $s_z$ denotes $\sin(\theta_z)$. $c_x$, $s_x$, $c_y$ *and* $s_y$ are similarly defined. From the above equation, Euler Angles can be derived from a rotation matrix by inverse trigonometric computation described as follows.

$$\theta_x = \arcsin(m_7)$$

$$\theta_z = \arctan(-\frac{m_1}{m_4})$$

$$\theta_y = \arctan(-\frac{m_6}{m_8})$$

Note, there are infinite solutions to the above equations which means that many sets of Euler Angles will result in the same rotation. Usually, the correct solution is selected according to the previous frame. It is expected that there is no big jump for a joint orientation between two sequential frames in a natural and smooth motion.

A quaternion has 4 components and typically is denoted as an ordered pair of a real number and a vector: $\mathbf{q} = (w, \mathbf{v}) = w + x\mathbf{i} + y\mathbf{j} + z\mathbf{k}$. When a unit quaternion ($w^2 + x^2 + y^2 + z^2 = 1$) is used to represent a 3D rotation, it specifies a rotation around a axis vector by $\theta$ angle, where the real part represents the $\cos(\frac{\theta}{2})$, the vector part represents the axis vector times $\sin(\frac{\theta}{2})$ (Equation B.1).

$$\mathbf{q} = \left(\cos\left(\frac{\theta}{2}\right), \sin\left(\frac{\theta}{2}\right)\mathbf{v}\right) \tag{B.1}$$

The product of two quaternion $\mathbf{q}_1$ and $\mathbf{q}_2$ is given by:

$$\mathbf{q}_1\mathbf{q}_2 = (w, \mathbf{v}) = (w_1, \mathbf{v}_1)(w_2, \mathbf{v}_2)$$

where

$$w = w_1 w_2 - \mathbf{v}_1\mathbf{v}_2$$

$$\mathbf{v} = w_1\mathbf{v}_2 + w_2\mathbf{v}_1 + \mathbf{v}_1 \times \mathbf{v}_2$$

The inverse of a quaternion $\mathbf{q}$ is:

$$\mathbf{q}^{-1} = \frac{(w,\ (-x,\ -y,\ -z)\ )}{(w^2 + x^2 + y^2 + z^2)}$$

From the interpretation of quaternion rotation, the quaternion forms of rotations around each axis are represented as follows.

$$\mathbf{q}_z = \left(\cos\left(\frac{\theta_z}{2}\right), \sin\left(\frac{\theta_z}{2}\right)(0,0,1)\right)$$
$$\mathbf{q}_x = \left(\cos\left(\frac{\theta_x}{2}\right), \sin\left(\frac{\theta_x}{2}\right)(1,0,0)\right)$$
$$\mathbf{q}_y = \left(\cos\left(\frac{\theta_y}{2}\right), \sin\left(\frac{\theta_y}{2}\right)(0,1,0)\right)$$

Therefore, the conversion from Euler Angles to quaternions is very straightforward. To express the orientation of a set of Euler Angles, the rotations around each axis are composed by multiplying the above 3 quaternions as follows.

$$\mathbf{q} = (w, (x,\ y,\ z)\ ) = \mathbf{q}_z\mathbf{q}_x\mathbf{q}_y$$

$$w = \cos\left(\frac{\theta_x}{2}\right)\cos\left(\frac{\theta_y}{2}\right)\cos\left(\frac{\theta_z}{2}\right) - \sin\left(\frac{\theta_x}{2}\right)\sin\left(\frac{\theta_y}{2}\right)\sin\left(\frac{\theta_z}{2}\right)$$
$$x = \sin\left(\frac{\theta_x}{2}\right)\cos\left(\frac{\theta_y}{2}\right)\cos\left(\frac{\theta_z}{2}\right) - \cos\left(\frac{\theta_x}{2}\right)\sin\left(\frac{\theta_y}{2}\right)\sin\left(\frac{\theta_z}{2}\right)$$
$$y = \cos\left(\frac{\theta_x}{2}\right)\sin\left(\frac{\theta_y}{2}\right)\cos\left(\frac{\theta_z}{2}\right) + \sin\left(\frac{\theta_x}{2}\right)\cos\left(\frac{\theta_y}{2}\right)\sin\left(\frac{\theta_z}{2}\right)$$
$$z = \sin\left(\frac{\theta_x}{2}\right)\sin\left(\frac{\theta_y}{2}\right)\cos\left(\frac{\theta_z}{2}\right) + \cos\left(\frac{\theta_x}{2}\right)\cos\left(\frac{\theta_y}{2}\right)\sin\left(\frac{\theta_z}{2}\right)$$